# Moo in practice - System::Image::Update

Jens Rehsack

Niederrhein Perl Mongers

2015

# Overview

## Part I

## Introduction

# Motivation

## Moo and System::Image::Update

- real world examples over far-fetched conceptuals
- MooX::ConfigFromFile and MooX::Options provide way more features and flexibility than MooseX competitors
- $2^{nd}$ generation of modern OO in Perl5

# Motivation

## Moo and System::Image::Update

- real world examples over far-fetched conceptuals
- MooX::ConfigFromFile and MooX::Options provide way more features and flexibility than MooseX competitors
- $2^{nd}$ generation of modern OO in Perl5

## System::Image::Update

- Provides an out-of-the-box solution for managing updates on embedded devices
- Easy to re-use in several layers of the firmware
  - self-sustaining (automatic) update management including forced updates (mind heartbleed)
  - embedable into middleware
  - ability for shortcuts
- self-healing capabilities

# Audience

## Audience

- Developer who wants to create or improve Perl5 software
- Developer who wants to learn how to develop modern OO with Perl5
- Developer who interested in developing mathematical exercises

# Audience

## Audience

- Developer who wants to create or improve Perl5 software
- Developer who wants to learn how to develop modern OO with Perl5
- Developer who interested in developing mathematical exercises

## Prerequisites of the Audience

Following knowledge is expected:

- General knowledge about object oriented programming or concepts like
  - classes
  - objects
  - polymorphism, inheritance and/or roles
  - methods, class functions
  - attributes, properties
- slightly above basic Perl experience

# Audience

## Audience

- Developer who wants to create or improve Perl5 software
- Developer who wants to learn how to develop modern OO with Perl5
- Developer who interested in developing mathematical exercises

## Prerequisites of the Audience

Following knowledge is expected:

- General knowledge about object oriented programming or concepts like
  - classes
  - objects
  - polymorphism, inheritance and/or roles
  - methods, class functions
  - attributes, properties
- slightly above basic Perl experience
- ever heard of Smalltalk and its OO-concept is a strong bonus

# Part II

# Moo basics

## Classes in Moo

- classes can be instantiated

## Classes in Moo

- classes can be instantiated
- one can inherit from classes

## Classes in Moo

- classes can be instantiated
- one can inherit from classes
- one can aggregate classes

## Classes in Moo

- classes can be instantiated
- one can inherit from classes
- one can aggregate classes
- distinguish naming conflicts is up to developer

## Classes in Moo

- classes can be instantiated
- one can inherit from classes
- one can aggregate classes
- distinguish naming conflicts is up to developer

```
{
package System::Image::Update;

use Moo;
use MooX::Options with_config_from_file => 1;
use IO::Async ();
...
use File::ConfigDir::System::Image::Update qw(system_image_update_dir);
around BUILDARGS => sub {...};
sub run {..}
sub collect_savable_config {}
sub reset_config {}
sub save_config {}
}
System::Image::Update->new_with_options->run;
```

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)
- roles can be composed into classes

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)
- roles can be composed into classes
- one can't inherit from roles - only consume

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)
- roles can be composed into classes
- one can't inherit from roles - only consume
- roles cannot exist stand-alone

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)
- roles can be composed into classes
- one can't inherit from roles - only consume
- roles cannot exist stand-alone
- roles are consumed once

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)
- roles can be composed into classes
- one can't inherit from roles - only consume
- roles cannot exist stand-alone
- roles are consumed once
- naming conflicts cause compile time error

## Roles in Moo

- roles describe a dedicated behavior (e.g. `logger`)
- roles can be composed into classes
- one can't inherit from roles - only consume
- roles cannot exist stand-alone
- roles are consumed once
- naming conflicts cause compile time error

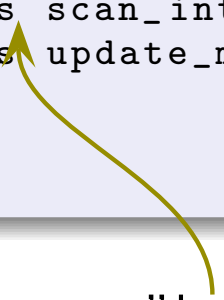
```
{   package System::Image::Update::Role::HTTP;

    use Moo::Role; # now it's a role - no 'is a' relationship anymore

    sub do_http_request { ... }
    around collect_savable_config => sub {...};
}
{   package System::Image::Update::Role::Scan;

    use Moo::Role;
    with "System::Image::Update::Role::HTTP"; # consumes a role

    sub scan { my $self = shift; $self->do_http_request(...) };
}
```

# Attributes in Moo

```
package System::Image::Update::Role::Scan;

use Moo::Role;

has scan_interval => ( is => "ro", default => 6*60*60 );
has update_manifest_uri => ( is => "lazy" );

1;
```

# Attributes in Moo

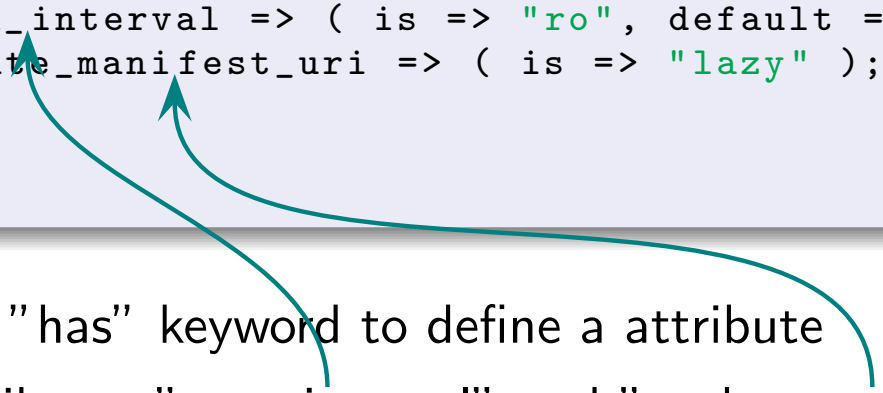
```
package System::Image::Update::Role::Scan;

use Moo::Role;

has scan_interval => ( is => "ro", default => 6*60*60 );
has update_manifest_uri => ( is => "lazy" );

1;
```



- use "has" keyword to define a attribute

## Attributes in Moo

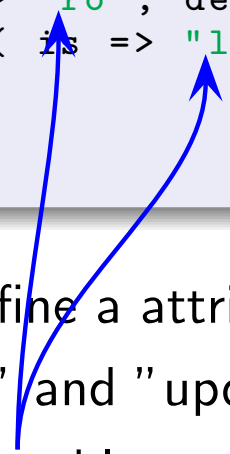
```
package System::Image::Update::Role::Scan;

use Moo::Role;

has scan_interval => ( is => "ro", default => 6*60*60 );
has update_manifest_uri => ( is => "lazy" );

1;
```



- use "has" keyword to define a attribute
- attributes "scan_interval" and "update_manifest_uri"

## Attributes in Moo

```
package System::Image::Update::Role::Scan;

use Moo::Role;

has scan_interval => ( is => "ro", default => 6*60*60 );
has update_manifest_uri => ( is => "lazy" );

1;
```



- use "has" keyword to define a attribute
- attributes "scan_interval" and "update_manifest_uri"
- those attributes are immutable

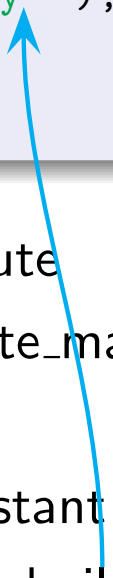
## Attributes in Moo

```
package System::Image::Update::Role::Scan;

use Moo::Role;

has scan_interval => ( is => "ro", default => 6*60*60 );
has update_manifest_uri => ( is => "lazy" );

1;
```

- use "has" keyword to define a attribute
- attributes "scan_interval" and "update_manifest_uri"
- those attributes are immutable
- scan_interval is initialized with a constant

## Attributes in Moo

```
package System::Image::Update::Role::Scan;

use Moo::Role;

has scan_interval => ( is => "ro", default => 6*60*60 );
has update_manifest_uri => ( is => "lazy" );

1;
```



- use "has" keyword to define a attribute
- attributes "scan_interval" and "update_manifest_uri"
- those attributes are immutable
- scan_interval is initialized with a constant
- update_manifest_uri is initialized by a builder

# Attribute options - Selection I

is **required** behavior description

## Attribute options - Selection I

is **required** behavior description

ro defines the attribute is read only

## Attribute options - Selection I

is **required** behavior description

- ro defines the attribute is read only
- rw defined the attribute is read-/writable

## Attribute options - Selection I

is **required** behavior description

| | |
|---|---|
| ro | defines the attribute is read only |
| rw | defined the attribute is read-/writable |
| lazy | defines the attribute is read only with a lazy initialization |

## Attribute options - Selection I

is **required** behavior description

- ro — defines the attribute is read only
- rw — defined the attribute is read-/writable
- lazy — defines the attribute is read only with a lazy initialization

required — when set to a true value, attribute must be passed on instantiation

## Attribute options - Selection I

is **required** behavior description

- ro — defines the attribute is read only
- rw — defined the attribute is read-/writable
- lazy — defines the attribute is read only with a lazy initialization

required — when set to a true value, attribute must be passed on instantiation

isa — defines a subroutine (`coderef`) which is called to validate values to set

## Attribute options - Selection I

is **required** behavior description

- ro — defines the attribute is read only
- rw — defined the attribute is read-/writable
- lazy — defines the attribute is read only with a lazy initialization

required — when set to a true value, attribute must be passed on instantiation

isa — defines a subroutine (`coderef`) which is called to validate values to set

coerce — defines a subroutine (`coderef`) which forces attribute values

## Attribute options - Selection I

is **required** behavior description

- ro — defines the attribute is read only
- rw — defined the attribute is read-/writable
- lazy — defines the attribute is read only with a lazy initialization

required — when set to a true value, attribute must be passed on instantiation

isa — defines a subroutine (`coderef`) which is called to validate values to set

coerce — defines a subroutine (`coderef`) which forces attribute values

trigger — takes a subroutine (`coderef`) which called anytime the attribute is set

## Attribute options - Selection I

is **required** behavior description

- ro: defines the attribute is read only
- rw: defined the attribute is read-/writable
- lazy: defines the attribute is read only with a lazy initialization

required: when set to a true value, attribute must be passed on instantiation

isa: defines a subroutine (`coderef`) which is called to validate values to set

coerce: defines a subroutine (`coderef`) which forces attribute values

trigger: takes a subroutine (`coderef`) which called anytime the attribute is set

**special**: the value of 1 means to generate a (`coderef`) which calles the method `_trigger_${attr_name}` (This is called *attribute shortcut*)

## Attribute options - Selection I

is **required** behavior description

- ro — defines the attribute is read only
- rw — defined the attribute is read-/writable
- lazy — defines the attribute is read only with a lazy initialization

required — when set to a true value, attribute must be passed on instantiation

isa — defines a subroutine (`coderef`) which is called to validate values to set

coerce — defines a subroutine (`coderef`) which forces attribute values

trigger — takes a subroutine (`coderef`) which called anytime the attribute is set
**special**: the value of 1 means to generate a (`coderef`) which calles the method `_trigger_${attr_name}` (This is called *attribute shortcut*)

default — subroutine (`coderef`) which is called to initialize an attribute

## Attribute options - Selection II

builder takes a method name (`string`) which is called to initialize an attribute (supports *attribute shortcut*)

## Attribute options - Selection II

builder takes a method name (`string`) which is called to initialize an attribute (supports *attribute shortcut*)

init_arg Takes the name of the key to look for at instantiation time of the object. A common use of this is to make an underscored attribute have a non-underscored initialization name. undef means that passing the value in on instantiation is ignored.

## Attribute options - Selection II

builder takes a method name (`string`) which is called to initialize an attribute (supports *attribute shortcut*)

init_arg Takes the name of the key to look for at instantiation time of the object. A common use of this is to make an underscored attribute have a non-underscored initialization name. undef means that passing the value in on instantiation is ignored.

clearer takes a method name (`string`) which will clear the attribute (supports *attribute shortcut*)

## Attribute options - Selection II

builder takes a method name (`string`) which is called to initialize an attribute (supports *attribute shortcut*)

init_arg Takes the name of the key to look for at instantiation time of the object. A common use of this is to make an underscored attribute have a non-underscored initialization name. undef means that passing the value in on instantiation is ignored.

clearer takes a method name (`string`) which will clear the attribute (supports *attribute shortcut*)

predicate takes a method name (`string`) which will return true if an attribute has a value (supports *attribute shortcut*)

# Methods in Moo

```
package System::Image::Update::Role::Async;

use IO::Async; use IO::Async::Loop;
use IO::Async::Timer::Absolute; use IO::Async::Timer::Countdown;

use Moo::Role;

has loop => ( is => "lazy", predicate => 1 );
sub _build_loop { return IO::Async::Loop->new() }

sub wakeup_at { my ( $self, $when, $cb_method ) = @_;
    my $timer;
    $self->loop->add($timer = IO::Async::Timer::Absolute->new(
        time      => $when,
        on_expire => sub { $self->$cb_method },
    ));
    $timer;
}
```

# Methods in Moo

```
package System::Image::Update::Role::Async;

use IO::Async; use IO::Async::Loop;
use IO::Async::Timer::Absolute; use IO::Async::Timer::Countdown;

use Moo::Role;

has loop => ( is => "lazy", predicate => 1 );
sub _build_loop { return IO::Async::Loop->new() }

sub wakeup_at { my ( $self, $when, $cb_method ) = @_;
    my $timer;
    $self->loop->add($timer = IO::Async::Timer::Absolute->new(
        time      => $when,
        on_expire => sub { $self->$cb_method },
    ));
    $timer;
}
```

- nothing like `MooseX::Declare` - pure Perl5 keywords are enough for plain methods

## Method Modifiers

Method modifiers are a convenient feature from the CLOS (Common Lisp Object System) world:

## Method Modifiers

Method modifiers are a convenient feature from the CLOS (Common Lisp Object System) world:

before `before method(s) => sub { ...}`
: before is called before the method it is modifying. Its return value is totally ignored.

## Method Modifiers

Method modifiers are a convenient feature from the CLOS (Common Lisp Object System) world:

before `before method(s) => sub { ...}`
: before is called before the method it is modifying. Its return value is totally ignored.

after `after method(s) => sub { ...}`
: after is called after the method it is modifying. Its return value is totally ignored.

## Method Modifiers

Method modifiers are a convenient feature from the CLOS (Common Lisp Object System) world:

**before** `before method(s) => sub { ...}`
before is called before the method it is modifying. Its return value is totally ignored.

**after** `after method(s) => sub { ...}`
after is called after the method it is modifying. Its return value is totally ignored.

**around** `around method(s) => sub { ...}`
around is called instead of the method it is modifying. The method you're overriding is passed as `coderef` in the first argument.

## Method Modifiers

Method modifiers are a convenient feature from the CLOS (Common Lisp Object System) world:

**before** `before method(s) => sub { ...}`
before is called before the method it is modifying. Its return value is totally ignored.

**after** `after method(s) => sub { ...}`
after is called after the method it is modifying. Its return value is totally ignored.

**around** `around method(s) => sub { ...}`
around is called instead of the method it is modifying. The method you're overriding is passed as `coderef` in the first argument.

- No support for `super`, `override`, `inner` or `augment`

## Method Modifiers - Advantages

- supersedes `$self->SUPER::foo(@_)` syntax

## Method Modifiers - Advantages

- supersedes `$self->SUPER::foo(@_)` syntax
- cleaner interface than SUPER

## Method Modifiers - Advantages

- supersedes `$self->SUPER::foo(@_)` syntax
- cleaner interface than SUPER
- allows multiple modifiers in single namespace

## Method Modifiers - Advantages

- supersedes `$self->SUPER::foo(@_)` syntax
- cleaner interface than SUPER
- allows multiple modifiers in single namespace
- also possible from within roles and not restricted to inheritance

## Method Modifiers - Advantages

- supersedes `$self->SUPER::foo(@_)` syntax
- cleaner interface than SUPER
- allows multiple modifiers in single namespace
- also possible from within roles and not restricted to inheritance
- ensures that inherited methods invocation happens right (mostly - remember around)

## Methods Modifiers - around avoid calling $orig

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";
around _build_config_prefix => sub { "sysimg_update" };
```

## Methods Modifiers - around avoid calling $orig

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";
around _build_config_prefix => sub { "sysimg_update" };
```

- captures control

## Methods Modifiers - around avoid calling $orig

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";
around _build_config_prefix => sub { "sysimg_update" };
```

- captures control
- receives responsibility

## Methods Modifiers - around avoid calling $orig

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";
around _build_config_prefix => sub { "sysimg_update" };
```

- captures control
- receives responsibility
- runtime of modified method completely eliminated

# Methods Modifiers - around modifying $orig return value

```
package System::Image::Update::Role::Scan;

use strict; use warnings; use Moo::Role;

around collect_savable_config => sub {
    my $next                    = shift;
    my $self                    = shift;
    my $collect_savable_config = $self->$next(@_);

    $self->update_server eq $default_update_server or $collect_savable_config->
    ...
    $collect_savable_config
}
```

# Methods Modifiers - around modifying $orig return value

```
package System::Image::Update::Role::Scan;

use strict; use warnings; use Moo::Role;

around collect_savable_config => sub {
    my $next                    = shift;
    my $self                    = shift;
    my $collect_savable_config = $self->$next(@_);

    $self->update_server eq $default_update_server or $collect_savable_config->
    ...
    $collect_savable_config
}
```

- modifies only required part

# Methods Modifiers - around modifying $orig return value

```
package System::Image::Update::Role::Scan;

use strict; use warnings; use Moo::Role;

around collect_savable_config => sub {
    my $next                  = shift;
    my $self                  = shift;
    my $collect_savable_config = $self->$next(@_);

    $self->update_server eq $default_update_server or $collect_savable_config->
    ...
    $collect_savable_config
}
```

- modifies only required part
- leaves most responsibility in modified method

# Methods Modifiers - around modifying $orig return value

```
package System::Image::Update::Role::Scan;

use strict; use warnings; use Moo::Role;

around collect_savable_config => sub {
    my $next                  = shift;
    my $self                  = shift;
    my $collect_savable_config = $self->$next(@_);

    $self->update_server eq $default_update_server or $collect_savable_config->
    ...
    $collect_savable_config
}
```

- modifies only required part
- leaves most responsibility in modified method
- runtime of modified method added to this method's runtime

# Part III

# System::Image::Update

# Rademacher Elektronik GmbH, Rhede

# Rademacher Elektronik GmbH, Rhede

- Successor of Guruplug based Homepilot 1

# Rademacher Elektronik GmbH, Rhede

- Successor of Guruplug based Homepilot 1
- Full update abilities (including boot loader, kernel and system components)

# Rademacher Elektronik GmbH, Rhede

- Successor of Guruplug based Homepilot 1
- Full update abilities (including boot loader, kernel and system components)
- Multimedia features (Mediaplayer with HDMI video and Coaxial audio)

## Rademacher Elektronik GmbH, Rhede

- Successor of Guruplug based Homepilot 1
- Full update abilities (including boot loader, kernel and system components)
- Multimedia features (Mediaplayer with HDMI video and Coaxial audio)
- Closer Adoption to Rademacher Way: Control from Hardware development until Customer Service

# HomePilot 2

# HomePilot 2


- service console moved from PHP to Perl5, PSGI and Dancer

# HomePilot 2


- service console moved from PHP to Perl5, PSGI and Dancer
- system management and automation full Perl5 powered

# HomePilot 2


- service console moved from PHP to Perl5, PSGI and Dancer
- system management and automation full Perl5 powered
- company infrastructure improved by Perl (eg. production of HP2)

# HomePilot 2

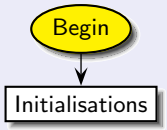

- service console moved from PHP to Perl5, PSGI and Dancer
- system management and automation full Perl5 powered
- company infrastructure improved by Perl (eg. production of HP2)
- PoC use Perl6 + NativeCall to eliminate wrapper processes

# HomePilot 2

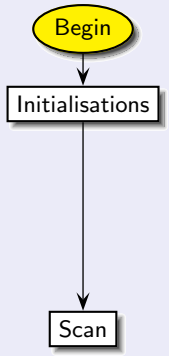

- service console moved from PHP to Perl5, PSGI and Dancer
- system management and automation full Perl5 powered
- company infrastructure improved by Perl (eg. production of HP2)
- PoC use Perl6 + NativeCall to eliminate wrapper processes
- created Yocto CPAN Layer for cross compiling lot's of CPAN modules

# State-Machine with togglable states and protected states

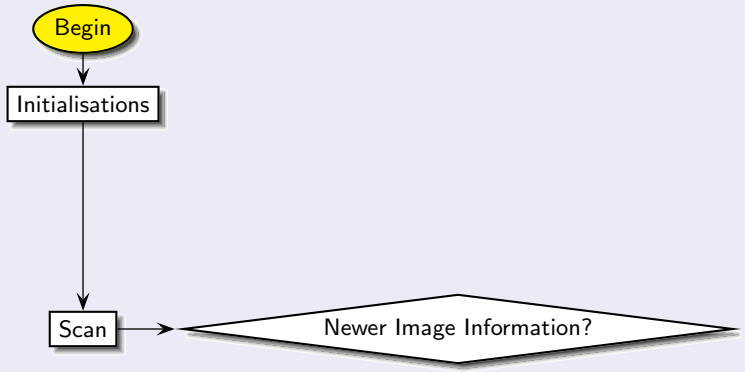
# State-Machine with togglable states and protected states



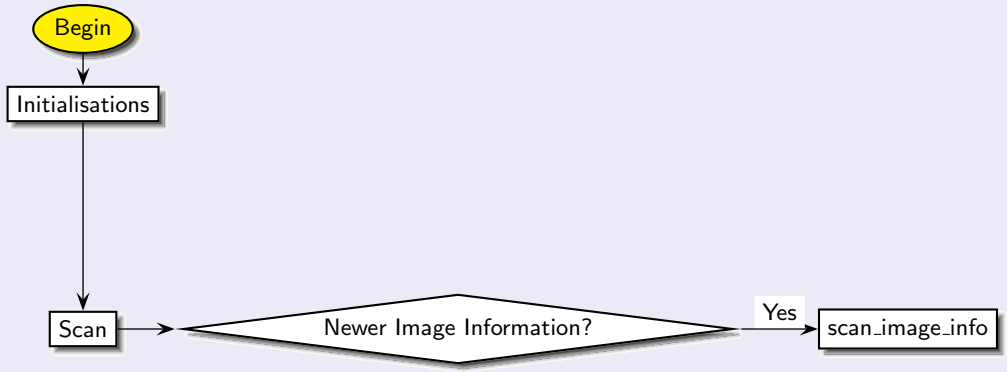
# State-Machine with togglable states and protected states



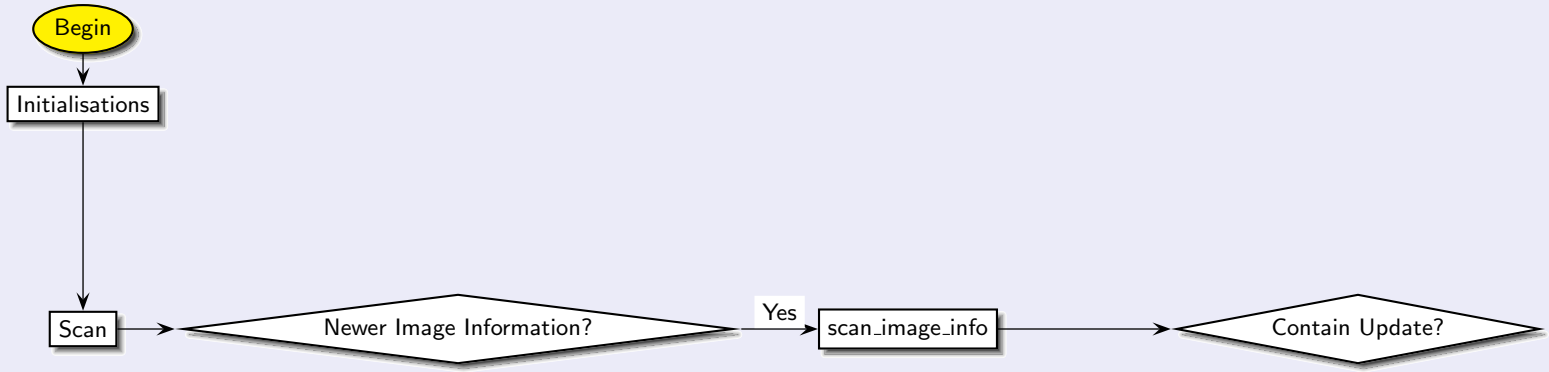
# State-Machine with togglable states and protected states



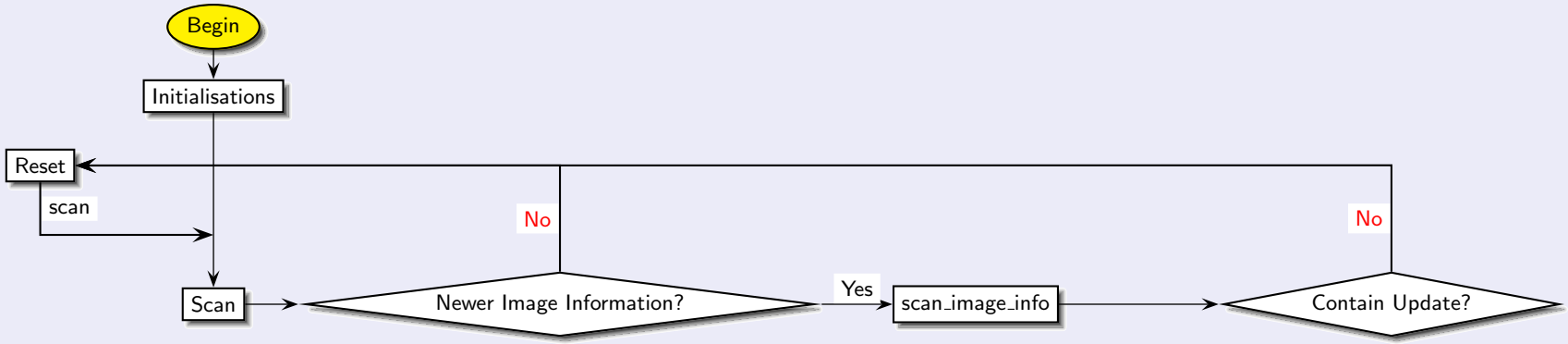
# State-Machine with togglable states and protected states



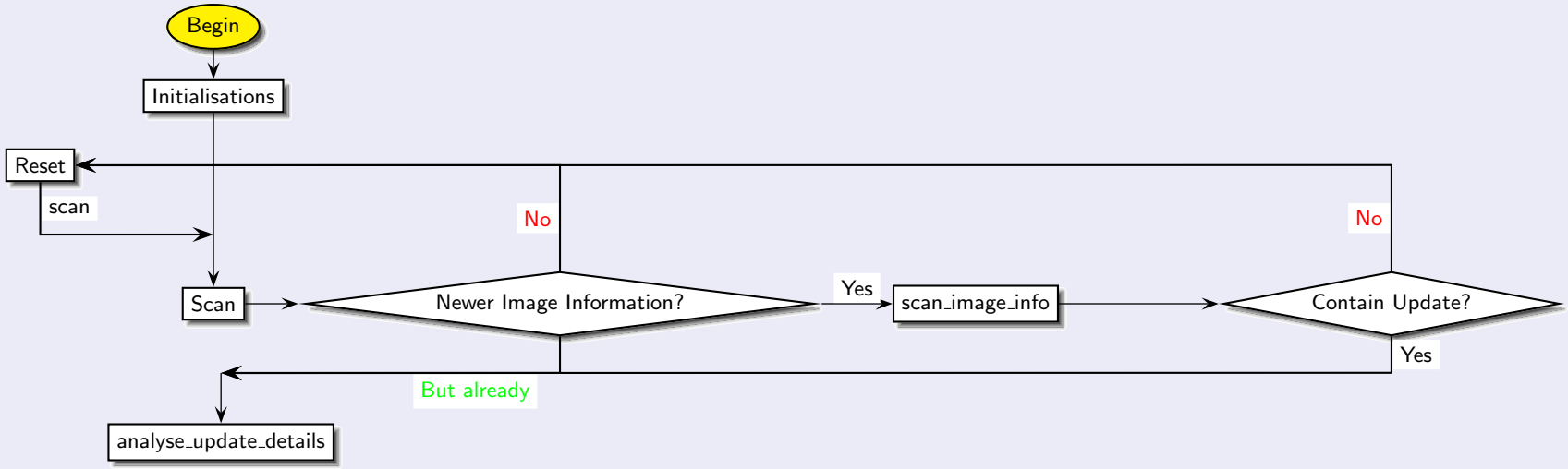
# State-Machine with togglable states and protected states



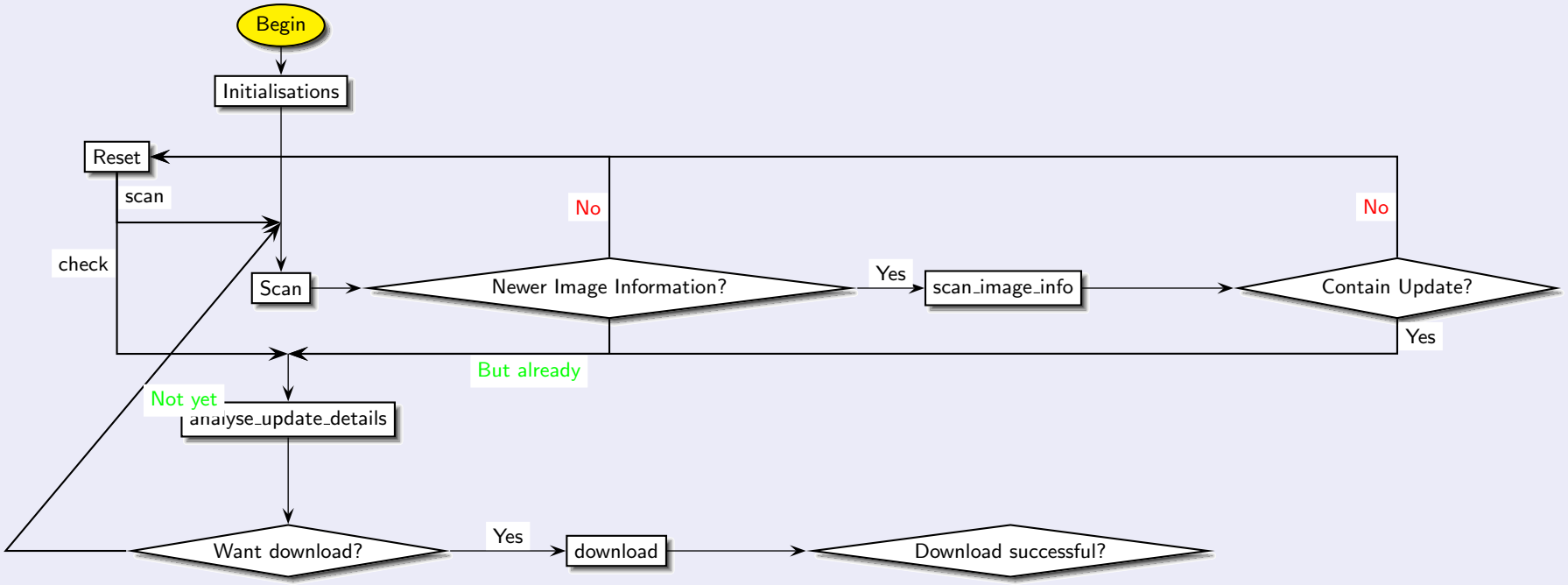
# State-Machine with togglable states and protected states



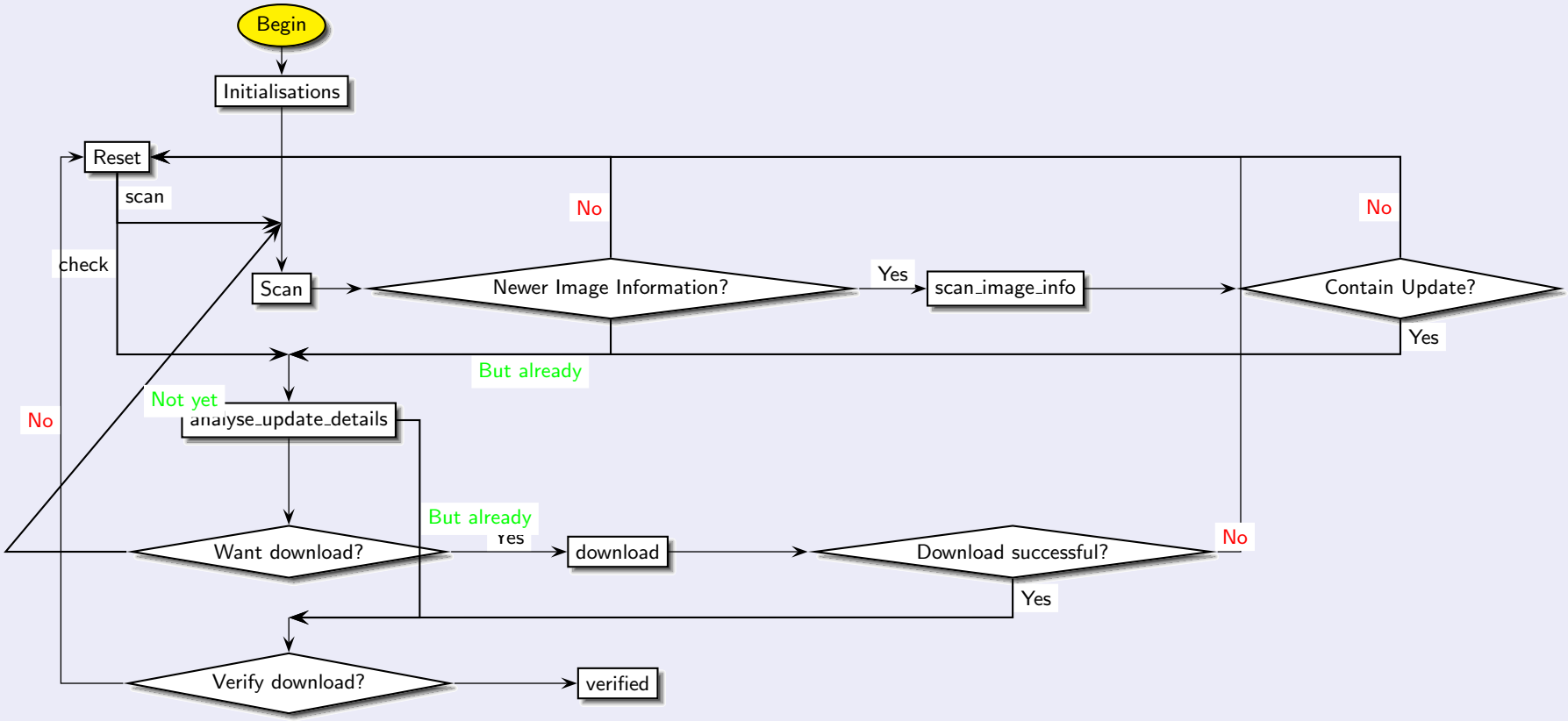
# State-Machine with togglable states and protected states



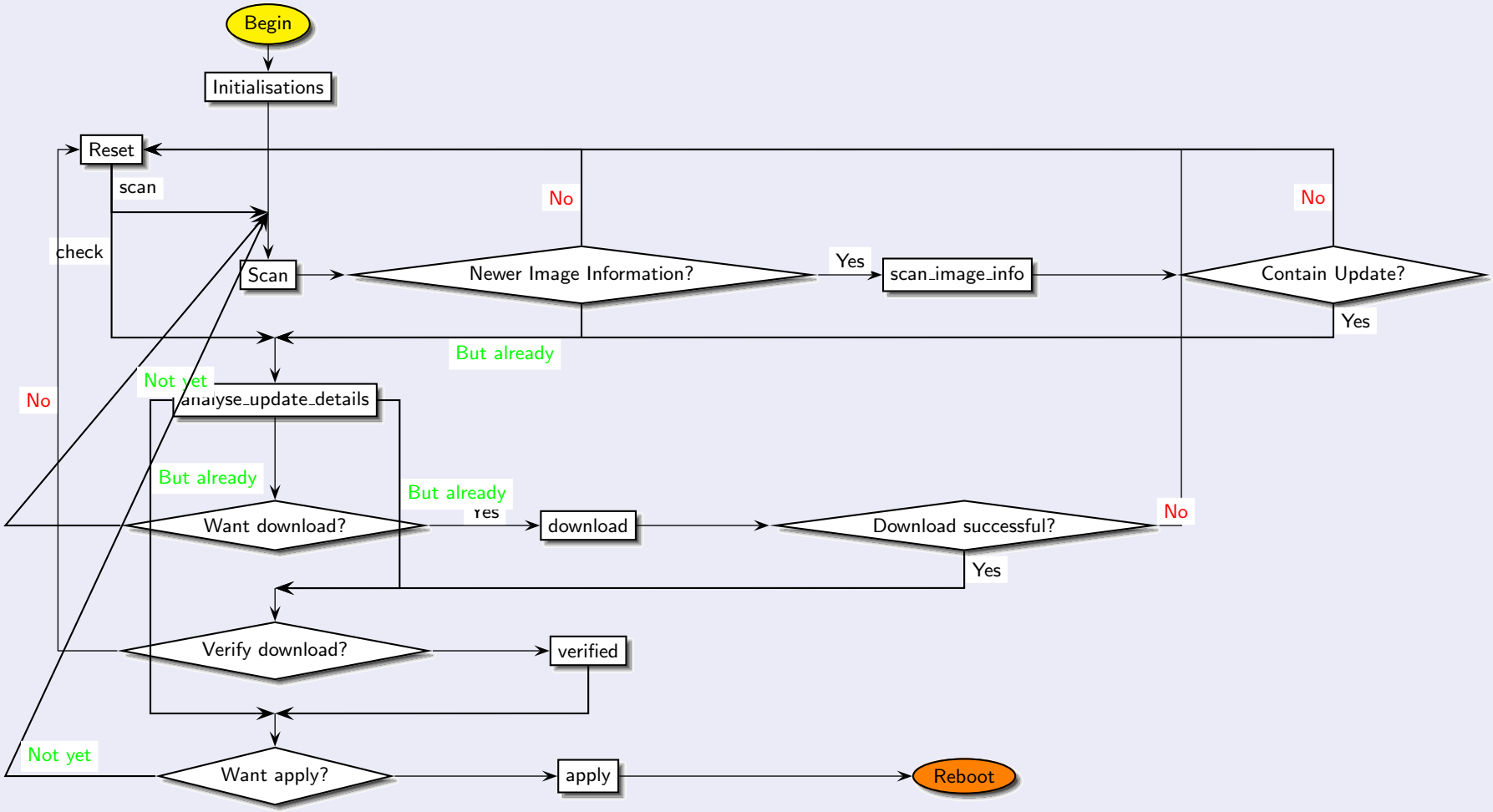
# State-Machine with togglable states and protected states



# State-Machine with togglable states and protected states



# State-Machine with togglable states and protected states



## State Control

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( is => "rw", lazy => 1, builder => 1, predicate => 1,
    isa => sub { __PACKAGE__->can( $_[0] ) or die "Invalid status: $_[0]" }
);

sub _build_status { -f $_[0]->update_manifest ? "check" :
    $_[0]->has_recent_update and -e $_[0]->download_image ? "prove" : "scan";
}
```

## State Control

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( is => "rw", lazy => 1, builder => 1, predicate => 1,
    isa => sub { __PACKAGE__->can( $_[0] ) or die "Invalid status: $_[0]" }
);

sub _build_status { -f $_[0]->update_manifest ? "check" :
    $_[0]->has_recent_update and -e $_[0]->download_image ? "prove" : "scan";
}
```

- automatic recovering after down-state (power outage, Vodka party, . . . )

## State Control

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( is => "rw", lazy => 1, builder => 1, predicate => 1,
    isa => sub { __PACKAGE__->can( $_[0] ) or die "Invalid status: $_[0]" }
);

sub _build_status { -f $_[0]->update_manifest ? "check" :
    $_[0]->has_recent_update and -e $_[0]->download_image ? "prove" : "scan";
}
```

- automatic recovering after down-state (power outage, Vodka party, . . . )
- room for improvements like continue aborted download

## State Control

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( is => "rw", lazy => 1, builder => 1, predicate => 1,
    isa => sub { __PACKAGE__->can( $_[0] ) or die "Invalid status: $_[0]" }
);

sub _build_status { -f $_[0]->update_manifest ? "check" :
    $_[0]->has_recent_update and -e $_[0]->download_image ? "prove" : "scan";
}
```

- automatic recovering after down-state (power outage, Vodka party, . . . )
- room for improvements like continue aborted download
- no direct path to "download" or "apply" to avoid mistakes

# State Control II

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( ... );

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);

    $params->{status} and $params->{status} eq "apply"
      and $params->{status} = "prove";
    $params->{status} and $params->{status} eq "prove"
      and $params->{recent_update}
      and $params->{recent_update}->{apply} = DateTime->now->epoch;

    $params;
};
```

## State Control II

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( ... );

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);

    $params->{status} and $params->{status} eq "apply"
      and $params->{status} = "prove";
    $params->{status} and $params->{status} eq "prove"
      and $params->{recent_update}
      and $params->{recent_update}->{apply} = DateTime->now->epoch;

    $params;
};
```

- toggleable are "download" and "prove"

## State Control II

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( ... );

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);

    $params->{status} and $params->{status} eq "apply"
      and $params->{status} = "prove";
    $params->{status} and $params->{status} eq "prove"
      and $params->{recent_update}
      and $params->{recent_update}->{apply} = DateTime->now->epoch;

    $params;
};
```

- toggleable are "download" and "prove"
- "apply" is protected by "prove" to ensure no corrupted image is applied

## State Control II

```
package System::Image::Update;

use strict; use warnings; use Moo;
with "System::Image::Update::Role::Scan", "System::Image::Update::Role::Check",

has status => ( ... );

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);

    $params->{status} and $params->{status} eq "apply"
      and $params->{status} = "prove";
    $params->{status} and $params->{status} eq "prove"
      and $params->{recent_update}
      and $params->{recent_update}->{apply} = DateTime->now->epoch;

    $params;
};
```

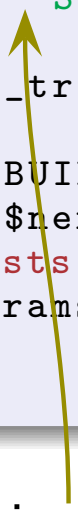
- toggleable are "download" and "prove"
- "apply" is protected by "prove" to ensure no corrupted image is applied
- protection needs to be improved before releasing to wildlife

# Middleware Information Center

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```

## Middleware Information Center

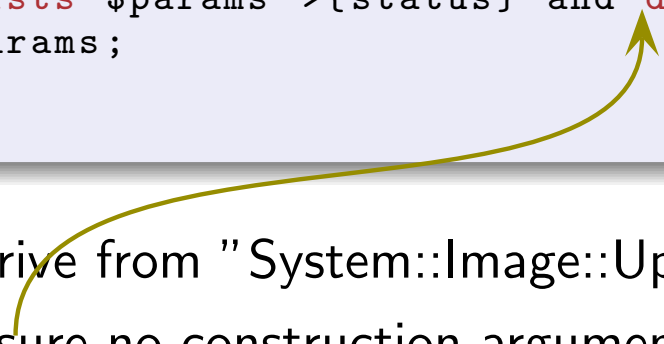
```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```



- derive from "System::Image::Update" to get the real world picture

## Middleware Information Center

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```



- derive from "System::Image::Update" to get the real world picture
- ensure no construction argument wastes the self-diagnostics

## Middleware Information Center

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```

- derive from "System::Image::Update" to get the real world picture
- ensure no construction argument wastes the self-diagnostics
- prevent automatism starts when recent update is found

## Middleware Information Center

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```

- derive from "System::Image::Update" to get the real world picture
- ensure no construction argument wastes the self-diagnostics
- prevent automatism starts when recent update is found

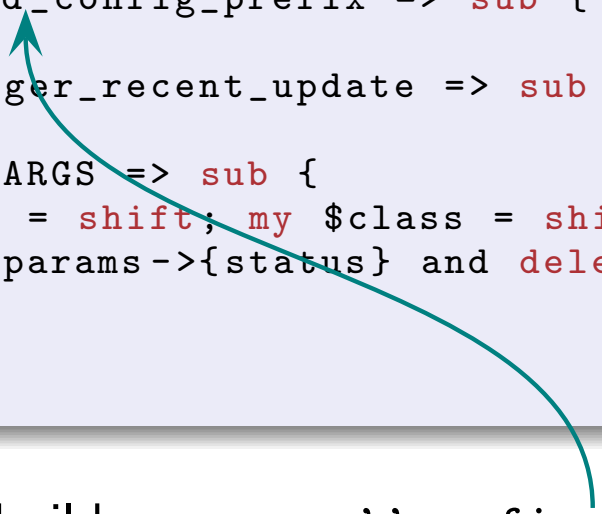
What information?

# Middleware Information Center

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _build_config_prefix => sub { "sysimg_update" };

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```

# Middleware Information Center

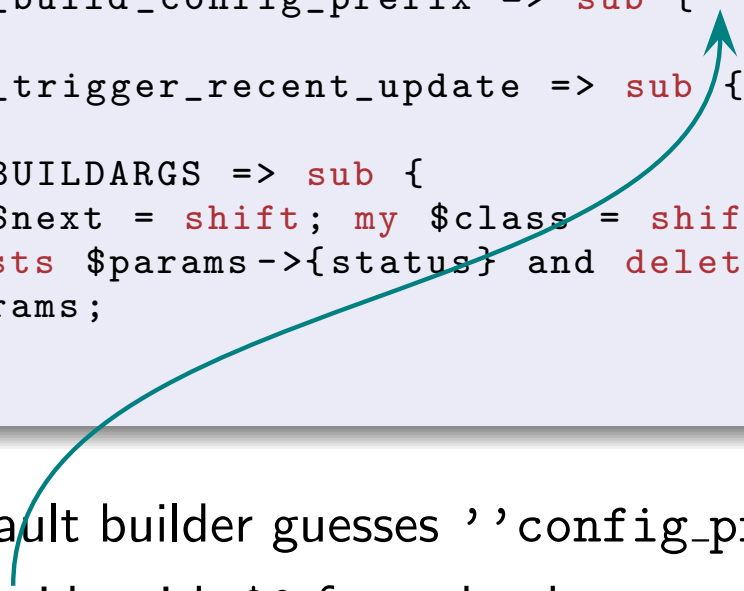
```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _build_config_prefix => sub { "sysimg_update" };

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```



- default builder guesses ''config_prefix'' from $0

## Middleware Information Center

```
package Update::Status;

use strict; use warnings; use Moo;
extends "System::Image::Update";

around _build_config_prefix => sub { "sysimg_update" };

around _trigger_recent_update => sub {};

around BUILDARGS => sub {
    my $next = shift; my $class = shift; my $params = $class->$next(@_);
    exists $params->{status} and delete $params->{status};
    $params;
};
```



- default builder guesses ''config_prefix'' from $0
- override with $0 from the daemon

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest

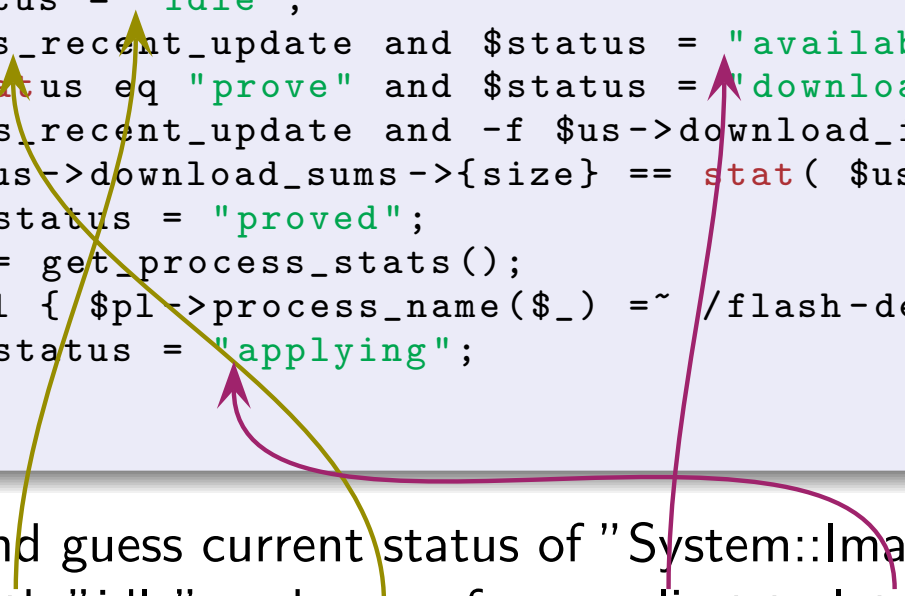
# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```



- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest
- better status overrides earlier measures - ''prove'' implies has_recent_update

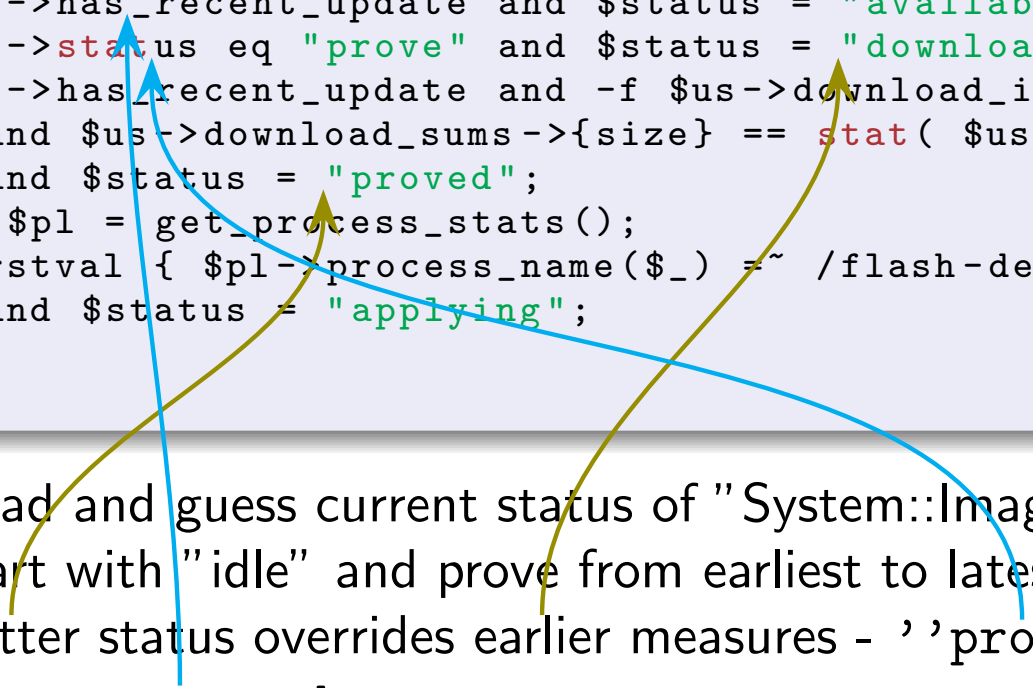
# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest
- better status overrides earlier measures - ''prove'' implies has_recent_update

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```



- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest
- better status overrides earlier measures - ''prove'' implies has_recent_update

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries()
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest
- better status overrides earlier measures - ''prove'' implies has_recent_update
- ''apply'' is currently done by an external process

# Middleware Delivery Center

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

get '/status' => sub {
    my $us = Update::Status->new;
    my $status = "idle";
    $us->has_recent_update and $status = "available";
    $us->status eq "prove" and $status = "downloading";
    $us->has_recent_update and -f $us->download_image
      and $us->download_sums->{size} == stat( $us->download_image )->size
      and $status = "proved";
    my $pl = get_process_stats();
    firstval { $pl->process_name($_) =~ /flash-device/ } ( 0 .. $pl->entries() )
      and $status = "applying";
};
```

- Load and guess current status of "System::Image::Update" instance
- start with "idle" and prove from earliest to latest
- better status overrides earlier measures - ''prove'' implies has_recent_update
- ''apply'' is currently done by an external process
- use Unix::Statgrab to grep for processes

# Middleware Strikes Back

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

put '/status/downloading' => sub {
    my $us = Update::Status->new();
    $us->has_recent_update or return $json->encode( { result => "n/a" } );
    $us->status("download"); $us->save_config;
    system("svc -t /etc/daemontools/service/sysimg_update/");
    return $json->encode( { result => "ok" } );
};
```

## Middleware Strikes Back

```
package hp2sm;
use strict; use warnings; use Dancer2 ':syntax'; ...; use Update::Status;

put '/status/downloading' => sub {
    my $us = Update::Status->new();
    $us->has_recent_update or return $json->encode( { result => "n/a" } );
    $us->status("download"); $us->save_config;
    system("svc -t /etc/daemontools/service/sysimg_update/");
    return $json->encode( { result => "ok" } );
};
```

## Middleware Information Boosted Persistency

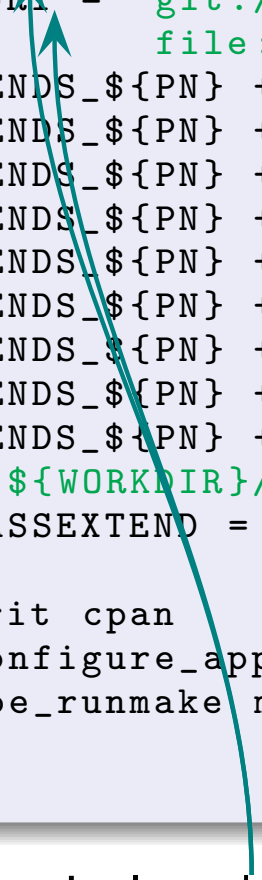
```
package Update::Status;
use strict; use warnings; use Moo;
extends "System::Image::Update";

around collect_savable_config => sub {
    my $next = shift; my $self = shift; my $save_cfg = $self->$next(@_);
    $self->has_status and $save_cfg->{status} = $self->status;
    $self->has_download_file
      and $save_cfg->{download_file} = $self->download_file;
    $save_cfg; };
```

## system-image-update_git.bb top

```
DESCRIPTION = "System::Image::Update helps managing updates of OS images ..."
SRC_URI = "git://github.com/rehsack/System-Image-Update.git;rev=646fa928... \
           file://run file://sysimg_update.json"
RDEPENDS_${PN} += "archive-peek-libarchive-perl crypt-ripemd160-perl"
RDEPENDS_${PN} += "datetime-format-strptime-perl"
RDEPENDS_${PN} += "log-any-adapter-dispatch-perl"
RDEPENDS_${PN} += "file-configdir-system-image-update-perl"
RDEPENDS_${PN} += "moo-perl moox-configfromfile-perl moox-log-any-perl"
RDEPENDS_${PN} += "moox-options-perl net-async-http-perl"
RDEPENDS_${PN} += "digest-md5-perl digest-md6-perl"
RDEPENDS_${PN} += "digest-sha-perl digest-sha3-perl"
RDEPENDS_${PN} += "daemontools system-image"
S = "${WORKDIR}/git"
BBCLASSEXTEND = "native"

inherit cpan
do_configure_append() {
    oe_runmake manifest
}
```

## system-image-update_git.bb top

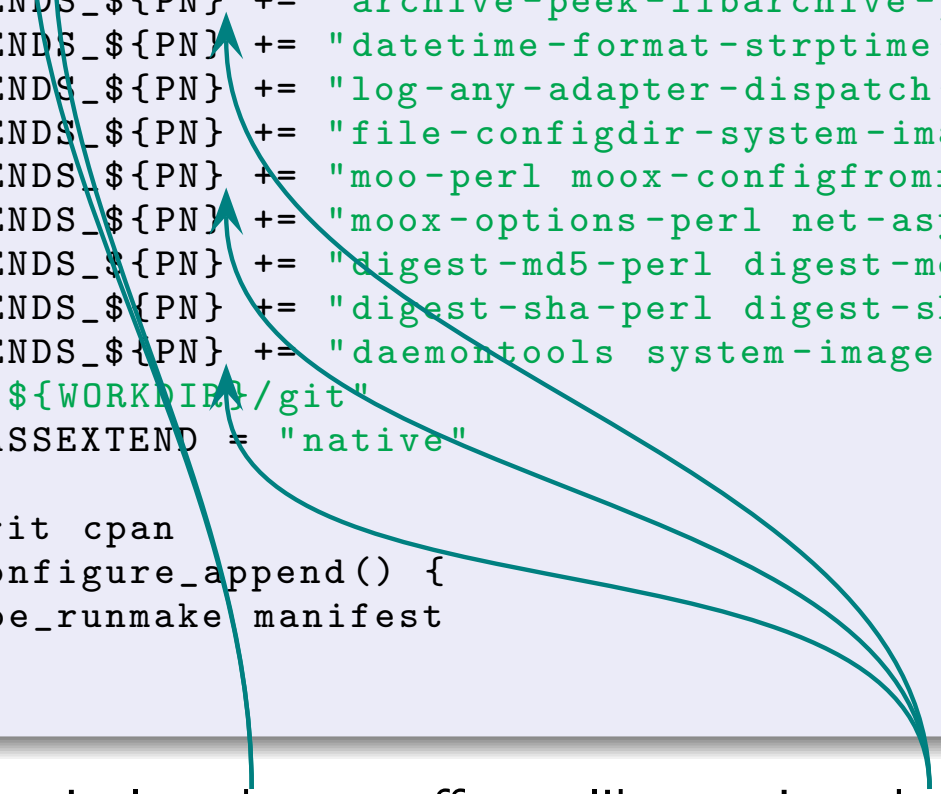
```
DESCRIPTION = "System::Image::Update helps managing updates of OS images ..."
SRC_URI = "git://github.com/rehsack/System-Image-Update.git;rev=646fa928... \
           file://run file://sysimg_update.json"
RDEPENDS_${PN} += "archive-peek-libarchive-perl crypt-ripemd160-perl"
RDEPENDS_${PN} += "datetime-format-strptime-perl"
RDEPENDS_${PN} += "log-any-adapter-dispatch-perl"
RDEPENDS_${PN} += "file-configdir-system-image-update-perl"
RDEPENDS_${PN} += "moo-perl moox-configfromfile-perl moox-log-any-perl"
RDEPENDS_${PN} += "moox-options-perl net-async-http-perl"
RDEPENDS_${PN} += "digest-md5-perl digest-md6-perl"
RDEPENDS_${PN} += "digest-sha-perl digest-sha3-perl"
RDEPENDS_${PN} += "daemontools system-image"
S = "${WORKDIR}/git"
BBCLASSEXTEND = "native"

inherit cpan
do_configure_append() {
    oe_runmake manifest
}
```



- typical package stuff . . . ,

## system-image-update_git.bb top

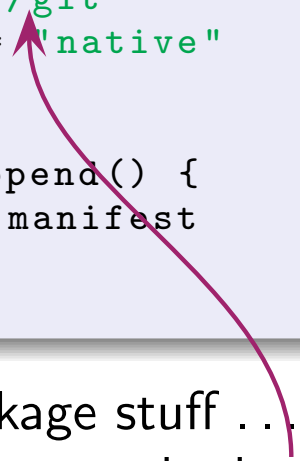
```
DESCRIPTION = "System::Image::Update helps managing updates of OS images ..."
SRC_URI = "git://github.com/rehsack/System-Image-Update.git;rev=646fa928... \
           file://run file://sysimg_update.json"
RDEPENDS_${PN} += "archive-peek-libarchive-perl crypt-ripemd160-perl"
RDEPENDS_${PN} += "datetime-format-strptime-perl"
RDEPENDS_${PN} += "log-any-adapter-dispatch-perl"
RDEPENDS_${PN} += "file-configdir-system-image-update-perl"
RDEPENDS_${PN} += "moo-perl moox-configfromfile-perl moox-log-any-perl"
RDEPENDS_${PN} += "moox-options-perl net-async-http-perl"
RDEPENDS_${PN} += "digest-md5-perl digest-md6-perl"
RDEPENDS_${PN} += "digest-sha-perl digest-sha3-perl"
RDEPENDS_${PN} += "daemontools system-image"
S = "${WORKDIR}/git"
BBCLASSEXTEND = "native"

inherit cpan
do_configure_append() {
    oe_runmake manifest
}
```



- typical package stuff . . . , like runtime dependencies

## system-image-update_git.bb top

```
DESCRIPTION = "System::Image::Update helps managing updates of OS images ..."
SRC_URI = "git://github.com/rehsack/System-Image-Update.git;rev=646fa928... \
           file://run file://sysimg_update.json"
RDEPENDS_${PN} += "archive-peek-libarchive-perl crypt-ripemd160-perl"
RDEPENDS_${PN} += "datetime-format-strptime-perl"
RDEPENDS_${PN} += "log-any-adapter-dispatch-perl"
RDEPENDS_${PN} += "file-configdir-system-image-update-perl"
RDEPENDS_${PN} += "moo-perl moox-configfromfile-perl moox-log-any-perl"
RDEPENDS_${PN} += "moox-options-perl net-async-http-perl"
RDEPENDS_${PN} += "digest-md5-perl digest-md6-perl"
RDEPENDS_${PN} += "digest-sha-perl digest-sha3-perl"
RDEPENDS_${PN} += "daemontools system-image"
S = "${WORKDIR}/git"
BBCLASSEXTEND = "native"

inherit cpan
do_configure_append() {
    oe_runmake manifest
}
```



- typical package stuff . . . , like runtime dependencies
- git checkouts need adoption of source path

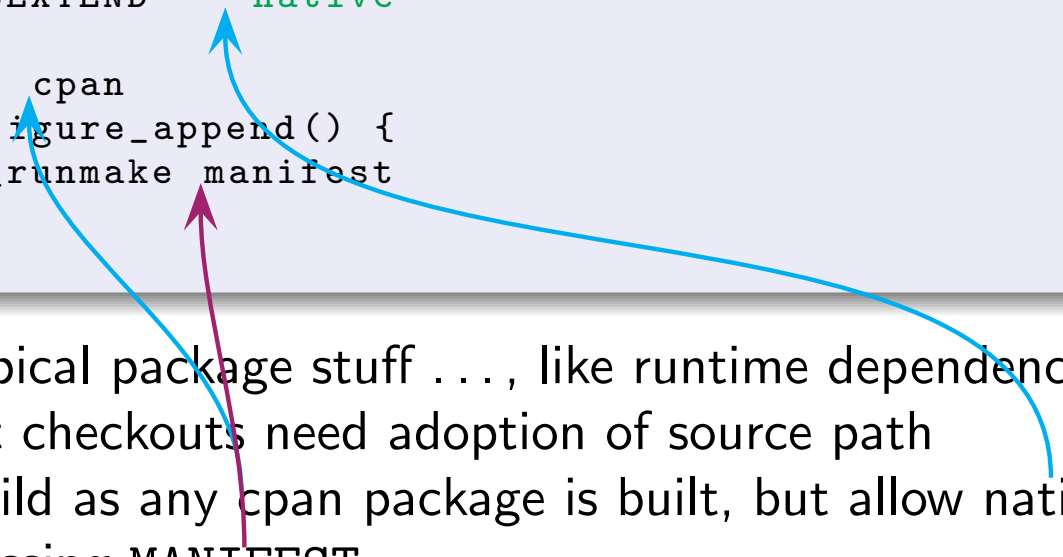
## system-image-update_git.bb top

```
DESCRIPTION = "System::Image::Update helps managing updates of OS images ..."
SRC_URI = "git://github.com/rehsack/System-Image-Update.git;rev=646fa928... \
           file://run file://sysimg_update.json"
RDEPENDS_${PN} += "archive-peek-libarchive-perl crypt-ripemd160-perl"
RDEPENDS_${PN} += "datetime-format-strptime-perl"
RDEPENDS_${PN} += "log-any-adapter-dispatch-perl"
RDEPENDS_${PN} += "file-configdir-system-image-update-perl"
RDEPENDS_${PN} += "moo-perl moox-configfromfile-perl moox-log-any-perl"
RDEPENDS_${PN} += "moox-options-perl net-async-http-perl"
RDEPENDS_${PN} += "digest-md5-perl digest-md6-perl"
RDEPENDS_${PN} += "digest-sha-perl digest-sha3-perl"
RDEPENDS_${PN} += "daemontools system-image"
S = "${WORKDIR}/git"
BBCLASSEXTEND = "native"

inherit cpan
do_configure_append() {
    oe_runmake manifest
}
```

- typical package stuff . . . , like runtime dependencies
- git checkouts need adoption of source path
- build as any cpan package is built, but allow native packages and create missing `MANIFEST`

## system-image-update_git.bb top

```
DESCRIPTION = "System::Image::Update helps managing updates of OS images ..."
SRC_URI = "git://github.com/rehsack/System-Image-Update.git;rev=646fa928... \
           file://run file://sysimg_update.json"
RDEPENDS_${PN} += "archive-peek-libarchive-perl crypt-ripemd160-perl"
RDEPENDS_${PN} += "datetime-format-strptime-perl"
RDEPENDS_${PN} += "log-any-adapter-dispatch-perl"
RDEPENDS_${PN} += "file-configdir-system-image-update-perl"
RDEPENDS_${PN} += "moo-perl moox-configfromfile-perl moox-log-any-perl"
RDEPENDS_${PN} += "moox-options-perl net-async-http-perl"
RDEPENDS_${PN} += "digest-md5-perl digest-md6-perl"
RDEPENDS_${PN} += "digest-sha-perl digest-sha3-perl"
RDEPENDS_${PN} += "daemontools system-image"
S = "${WORKDIR}/git"
BBCLASSEXTEND = "native"

inherit cpan
do_configure_append() {
    oe_runmake manifest
}
```



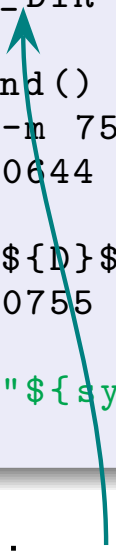
- typical package stuff . . . , like runtime dependencies
- git checkouts need adoption of source path
- build as any cpan package is built, but allow native packages and create missing `MANIFEST`

## system-image-update_git.bb bottom

```
SERVICE_ROOT = "${sysconfdir}/daemontools/service"
SYSUPDT_SERVICE_DIR = "${SERVICE_ROOT}/sysimg_update"

do_install_append() {
    install -d -m 755 ${D}${sysconfdir}
    install -m 0644 ${WORKDIR}/sysimg_update.json ${D}${sysconfdir}

    install -d ${D}${SYSUPDT_SERVICE_DIR}
    install -m 0755 ${WORKDIR}/run ${D}${SYSUPDT_SERVICE_DIR}/run
}
FILES_${PN} += "${sysconfdir}"
```

## system-image-update_git.bb bottom

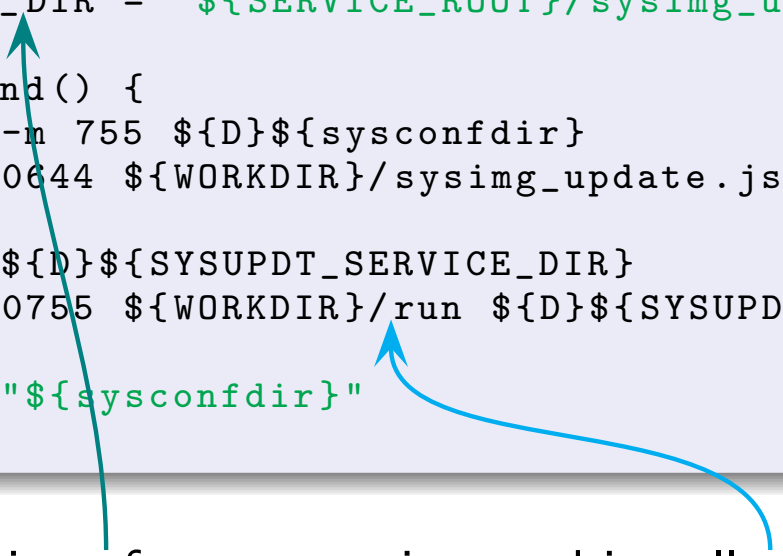
```
SERVICE_ROOT = "${sysconfdir}/daemontools/service"
SYSUPDT_SERVICE_DIR = "${SERVICE_ROOT}/sysimg_update"

do_install_append() {
    install -d -m 755 ${D}${sysconfdir}
    install -m 0644 ${WORKDIR}/sysimg_update.json ${D}${sysconfdir}

    install -d ${D}${SYSUPDT_SERVICE_DIR}
    install -m 0755 ${WORKDIR}/run ${D}${SYSUPDT_SERVICE_DIR}/run
}
FILES_${PN} += "${sysconfdir}"
```



- define location of startup scripts

## system-image-update_git.bb bottom

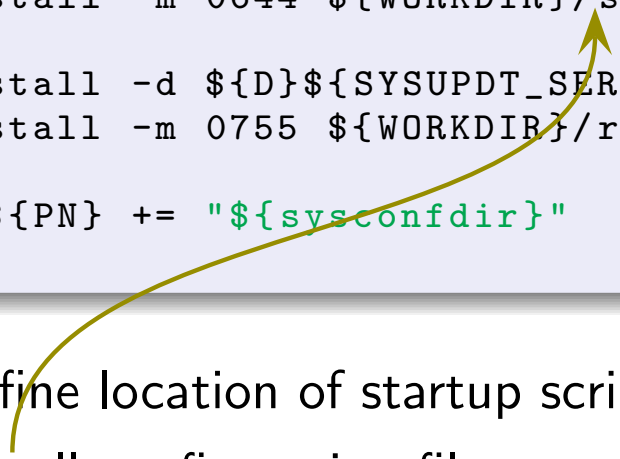
```
SERVICE_ROOT = "${sysconfdir}/daemontools/service"
SYSUPDT_SERVICE_DIR = "${SERVICE_ROOT}/sysimg_update"

do_install_append() {
    install -d -m 755 ${D}${sysconfdir}
    install -m 0644 ${WORKDIR}/sysimg_update.json ${D}${sysconfdir}

    install -d ${D}${SYSUPDT_SERVICE_DIR}
    install -m 0755 ${WORKDIR}/run ${D}${SYSUPDT_SERVICE_DIR}/run
}
FILES_${PN} += "${sysconfdir}"
```



- define location of startup scripts and install to there

## system-image-update_git.bb bottom

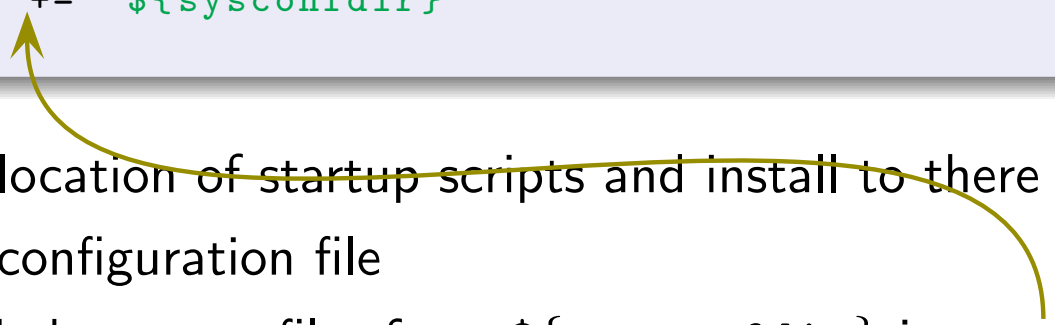
```
SERVICE_ROOT = "${sysconfdir}/daemontools/service"
SYSUPDT_SERVICE_DIR = "${SERVICE_ROOT}/sysimg_update"

do_install_append() {
    install -d -m 755 ${D}${sysconfdir}
    install -m 0644 ${WORKDIR}/sysimg_update.json ${D}${sysconfdir}

    install -d ${D}${SYSUPDT_SERVICE_DIR}
    install -m 0755 ${WORKDIR}/run ${D}${SYSUPDT_SERVICE_DIR}/run
}
FILES_${PN} += "${sysconfdir}"
```



- define location of startup scripts and install to there
- install configuration file

## system-image-update_git.bb bottom

```
SERVICE_ROOT = "${sysconfdir}/daemontools/service"
SYSUPDT_SERVICE_DIR = "${SERVICE_ROOT}/sysimg_update"

do_install_append() {
    install -d -m 755 ${D}${sysconfdir}
    install -m 0644 ${WORKDIR}/sysimg_update.json ${D}${sysconfdir}

    install -d ${D}${SYSUPDT_SERVICE_DIR}
    install -m 0755 ${WORKDIR}/run ${D}${SYSUPDT_SERVICE_DIR}/run
}
FILES_${PN} += "${sysconfdir}"
```



- define location of startup scripts and install to there
- install configuration file
- tell bitbake to put files from ${sysconfdir} into package

## sysimg_update.json

```
{
    "log_adapter" : [
        "Dispatch",
        "outputs", [
            [ "File",   "min_level", "debug", "filename",
              "/var/log/sysimg_update.log", "newline", 1, "mode", ">>" ],
            [ "File",   "min_level", "error", "filename",
              "/var/log/sysimg_update.error", "newline", 1, "mode", ">>" ],
            [ "Screen", "min_level", "notice", "newline", 1, "stderr", 1 ]
        ]
    ],
    "update_manifest_dirname" : "/rwmedia/update/",
    "http_user" : "b01f..."
}
```

## sysimg_update.json

```
{
    "log_adapter" : [
        "Dispatch",
        "outputs", [
            [ "File",   "min_level", "debug", "filename",
              "/var/log/sysimg_update.log", "newline", 1, "mode", ">>" ],
            [ "File",   "min_level", "error", "filename",
              "/var/log/sysimg_update.error", "newline", 1, "mode", ">>" ],
            [ "Screen", "min_level", "notice", "newline", 1, "stderr", 1 ]
        ]
    ],
    "update_manifest_dirname" : "/rwmedia/update/",
    "http_user" : "b01f..."
}
```

- Provides settings for `Log::Any` (mind `_trigger_log_adapter` in `System::Image::Update::Role::Logging` consuming `MooX::Log::Any`)

## sysimg_update.json

```
{
    "log_adapter" : [
        "Dispatch",
        "outputs", [
            [ "File",   "min_level", "debug", "filename",
              "/var/log/sysimg_update.log", "newline", 1, "mode", ">>" ],
            [ "File",   "min_level", "error", "filename",
              "/var/log/sysimg_update.error", "newline", 1, "mode", ">>" ],
            [ "Screen", "min_level", "notice", "newline", 1, "stderr", 1 ]
        ]
    ],
    "update_manifest_dirname" : "/rwmedia/update/",
    "http_user" : "b01f..."
}
```

- Provides settings for `Log::Any` (mind `_trigger_log_adapter` in `System::Image::Update::Role::Logging` consuming `MooX::Log::Any`)
- redirect place to store update manifest (files)

## sysimg_update.json

```
{
    "log_adapter" : [
        "Dispatch",
        "outputs", [
            [ "File",   "min_level", "debug", "filename",
              "/var/log/sysimg_update.log", "newline", 1, "mode", ">>" ],
            [ "File",   "min_level", "error", "filename",
              "/var/log/sysimg_update.error", "newline", 1, "mode", ">>" ],
            [ "Screen", "min_level", "notice", "newline", 1, "stderr", 1 ]
        ]
    ],
    "update_manifest_dirname" : "/rwmedia/update/",
    "http_user" : "b01f..."
}
```

- Provides settings for `Log::Any` (mind `_trigger_log_adapter` in `System::Image::Update::Role::Logging` consuming `MooX::Log::Any`)
- redirect place to store update manifest (files)
- Provide authentication to update server for development boxes (avoid `builder` is called)

# Overview

## Part IV

## Finish

## Conclusion

- lazy attributes allow designing a multi-stage initialization phase

## Conclusion

- lazy attributes allow designing a multi-stage initialization phase
- benefit of common runtime (faster load) when using

## Conclusion

- lazy attributes allow designing a multi-stage initialization phase
- benefit of common runtime (faster load) when using
- improve design by

## Conclusion

- lazy attributes allow designing a multi-stage initialization phase
- benefit of common runtime (faster load) when using
- improve design by
  - using roles for behavioral design (avoid duck typing)

## Conclusion

- lazy attributes allow designing a multi-stage initialization phase
- benefit of common runtime (faster load) when using
- improve design by
  - using roles for behavioral design (avoid duck typing)
  - using explicit patterns for clear separation of concerns

## Conclusion

- lazy attributes allow designing a multi-stage initialization phase
- benefit of common runtime (faster load) when using
- improve design by
  - using roles for behavioral design (avoid duck typing)
  - using explicit patterns for clear separation of concerns
  - express intensions clearer for method overloading by using *method modifiers*

# Resources

## Software on MetaCPAN

https://metacpan.org/pod/Moo
https://metacpan.org/search?q=MooX
https://metacpan.org/pod/MooX::Options
https://metacpan.org/pod/MooX::ConfigFromFile
https://metacpan.org/pod/IO::Async

# Resources

## Software on MetaCPAN

https://metacpan.org/pod/Moo
https://metacpan.org/search?q=MooX
https://metacpan.org/pod/MooX::Options
https://metacpan.org/pod/MooX::ConfigFromFile
https://metacpan.org/pod/IO::Async

## Software on GitHub

https://github.com/moose/Moo
https://github.com/rehsack/System-Image-Update
https://github.com/perl5-utils/File-ConfigDir-System-Image-Update

# Resources

## Software on MetaCPAN

https://metacpan.org/pod/Moo
https://metacpan.org/search?q=MooX
https://metacpan.org/pod/MooX::Options
https://metacpan.org/pod/MooX::ConfigFromFile
https://metacpan.org/pod/IO::Async

## Software on GitHub

https://github.com/moose/Moo
https://github.com/rehsack/System-Image-Update
https://github.com/perl5-utils/File-ConfigDir-System-Image-Update

## Software for Cross-Building Perl-Modules

https://www.yoctoproject.org/
https://github.com/rehsack/meta-cpan

# Resources

## IRC

irc://irc.perl.org/#moose
irc://irc.perl.org/#web-simple
irc://irc.perl.org/#dancer
irc://irc.freenode.org/#yocto

# Resources

## IRC

irc://irc.perl.org/#moose
irc://irc.perl.org/#web-simple
irc://irc.perl.org/#dancer
irc://irc.freenode.org/#yocto

## Hints

http://sck.pm/WVO # proper usage of the roles in perl
https://metacpan.org/pod/Moo#CLEANING-UP-IMPORTS

# Thank You For Listening

## Questions?

Jens Rehsack <rehsack@cpan.org>