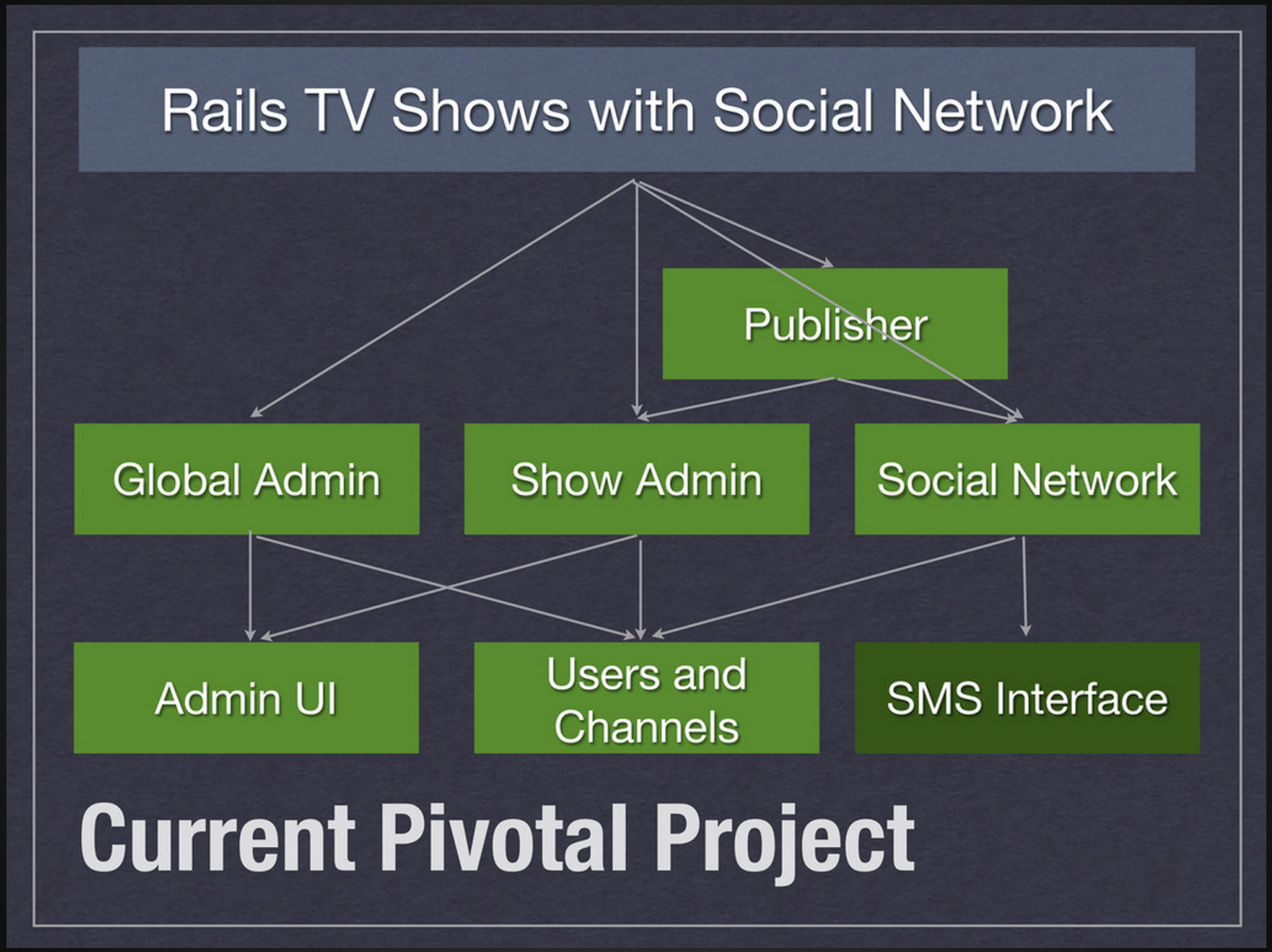
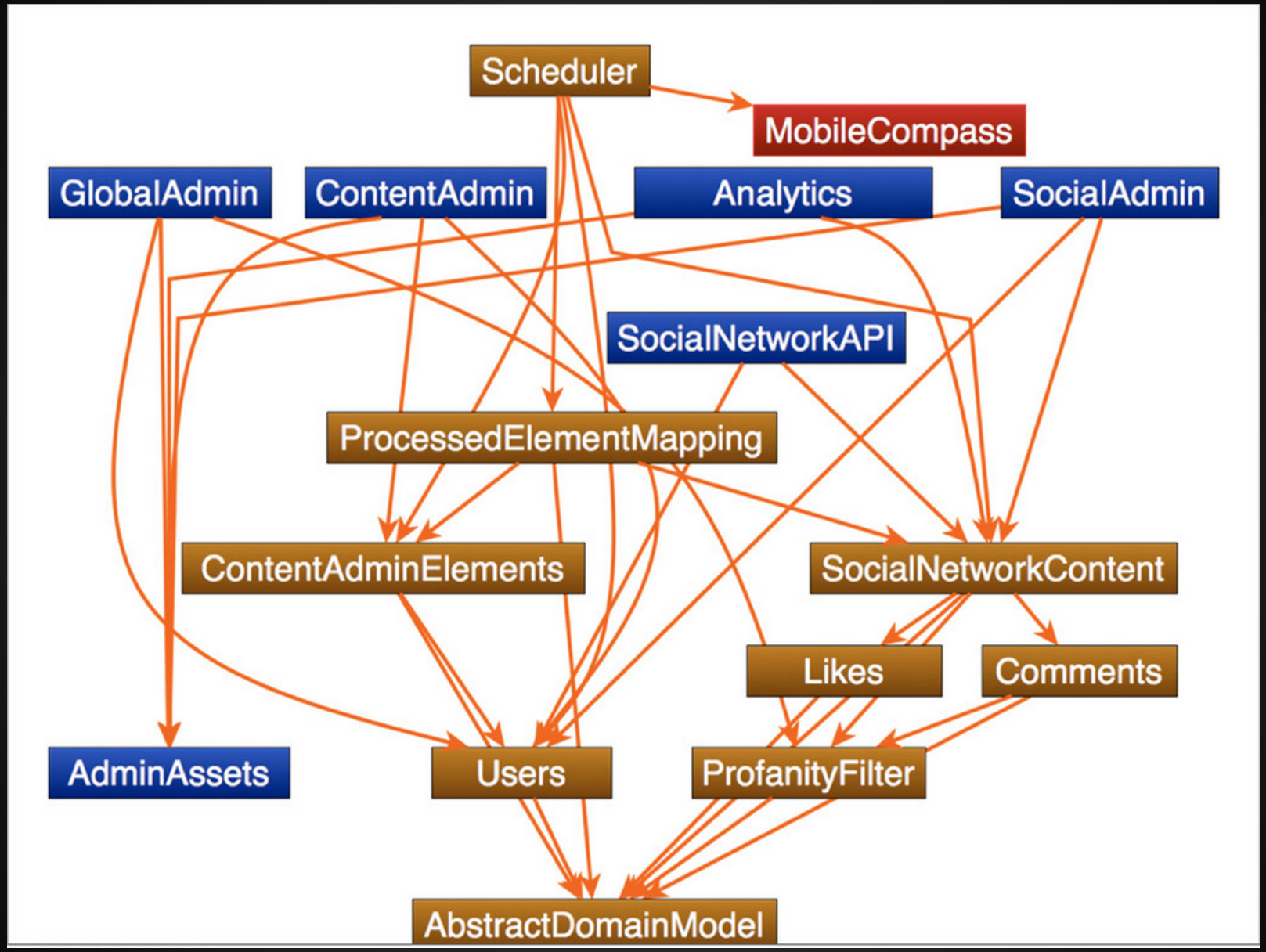
Active Record Model Dependencies

Stephan Hagemann

Lots of Architecture



https://speakerdeck.com/shagemann/wrangling-large-ruby-codebases



https://speakerdeck.com/benjaminleesmith/how-i-architected-my-big-rails-app-for-success-rmr-2013

Little bit of Architecture

The Pragmatic Programmers

Rails 4

Agile Web Development with Rails 4

Sam Ruby
Dave Thomas
David Heinemeier Hansson

Edited by Susannah Davidson Pfalzer

The Facets  of Ruby Series

declarations to our model files that specify their interrelationships. Open the newly created order.rb file in app/models, and add a call to has_many:

```
class Order < ActiveRecord::Base
  has_many :line_items
end
```

That has_many directive is fairly self-explanatory: an order (potentially) has many associated line items. These are linked to the order because each line item contains a reference to its order's id.

Now, for completeness, we should add a has_many directive to our product model. After all, if we have lots of orders, each product might have many line items referencing it.

```
class Product < ActiveRecord::Base
  has_many :line_items
  # ...
```

Next, we'll specify links in the opposite direction, from the line item to the orders and products tables. To do this, we use the belongs_to declaration twice in the line_item.rb file:

Download depot_p/app/models/line_item.rb

```
class LineItem < ActiveRecord::Base
  belongs_to :order
  belongs_to :product
end
```

belongs_to tells Rails that rows in the line_items table are children of rows in the orders and products tables. No line item can exist unless the corresponding order and product rows exist. There's an easy way to remember where to put belongs_to declarations: if a table has foreign keys, the corresponding model should have a belongs_to for each.

declarations to our model files that specify their interrelationships. Open the newly created order.rb file in app/models, and add a call to has_many:

That has_many directive is fairly self-explanatory: an order (potentially) has many associated line items. These are linked to the order because each line item contains a reference to its order's id.

item contains a reference to its order's id.

Now, for completeness, we should add a has_many directive to our product model. After all, if we have lots of orders, each product might have many line items referencing it.

```
class Product < ActiveRecord::Base
  has_many :line_items
  # ...
```

Next, we'll specify links in the opposite direction, from the line item to the orders and products tables. To do this, we use the belongs_to declaration twice in the line_item.rb file:

Download depot_p/app/models/line_item.rb

```
class LineItem < ActiveRecord::Base
  belongs_to :order
  belongs_to :product
end
```

belongs_to tells Rails that rows in the line_items table are children of rows in the orders and products tables. No line item can exist unless the corresponding order and product rows exist. There's an easy way to remember where to put belongs_to declarations: if a table has foreign keys, the corresponding model should have a belongs_to for each.

declarations to our model files that specify their interrelationships. Open the newly created order.rb file in app/models, and add a call to has_many:

That has_many directive is fairly self-explanatory: an order (potentially) has many associated line items. These are linked to the order because each line item contains a reference to its order's id.

item contains a reference to its order's id.

Now, for completeness, we should add a has_many directive to our product model. After all, if we have lots of orders, each product might have many line items referencing it.

```
  has_many :line_items
  # ...
```

Next, we'll specify links in the opposite direction, from the line item to the orders and products tables. To do this, we use the belongs_to declaration twice in the line_item.rb file:

Download depot_p/app/models/line_item.rb

```
class LineItem < ActiveRecord::Base
  belongs_to :order
  belongs_to :product
end
```

belongs_to tells Rails that rows in the line_items table are children of rows in the orders and products tables. No line item can exist unless the corresponding order and product rows exist. There's an easy way to remember where to put belongs_to declarations: if a table has foreign keys, the corresponding model should have a belongs_to for each.

declarations to our model files that specify their interrelationships. Open the newly created order.rb file in app/models, and add a call to has_many:

That has_many directive is fairly self-explanatory: an order (potentially) has many associated line items. These are linked to the order because each line item contains a reference to its order's id.

item contains a reference to its order's id.

Now, for completeness, we should add a has_many directive to our product model. After all, if we have lots of orders, each product might have many line items referencing it.

```
has_many :line_items
# ...
```

Next, we'll specify links in the opposite direction, from the line item to the orders and products tables. To do this, we use the belongs_to declaration twice in the line_item.rb file:

```
class LineItem < ActiveRecord::Base
  belongs_to :order
  belongs_to :product
end
```

belongs_to tells Rails that rows in the line_items table are children of rows in the orders and products tables. No line item can exist unless the corresponding order and product rows exist. There's an easy way to remember where to put belongs_to declarations: if a table has foreign keys, the corresponding model should have a belongs_to for each.

declarations to our model files that specify their interrelationships. Open the newly created order.rb file in app/models, and add a call to has_many:

That has_many directive is fairly self-explanatory: an order (potentially) has many associated line items. These are linked to the order because each line item contains a reference to its order's id.

item contains a reference to its order's id.

Now, for completeness, we should add a has_many directive to our product model. After all, if we have lots of orders, each product might have many line items referencing it.

```
has_many :line_items
# ...
```

Next, we'll specify links in the opposite direction, from the line item to the orders and products tables. To do this, we use the belongs_to declaration twice in the line_item.rb file:

```
class LineItem < ActiveRecord::Base
  belongs_to :order
  belongs_to :product
end
```

belongs_to tells Rails that rows in the line_items table are children of rows in order and product rows exist. There's an easy way to remember where to put belongs_to declarations: if a table has foreign keys, the corresponding model should have a belongs_to for each.

What does that lead to?

Unclear responsibilities

MRP not SRP

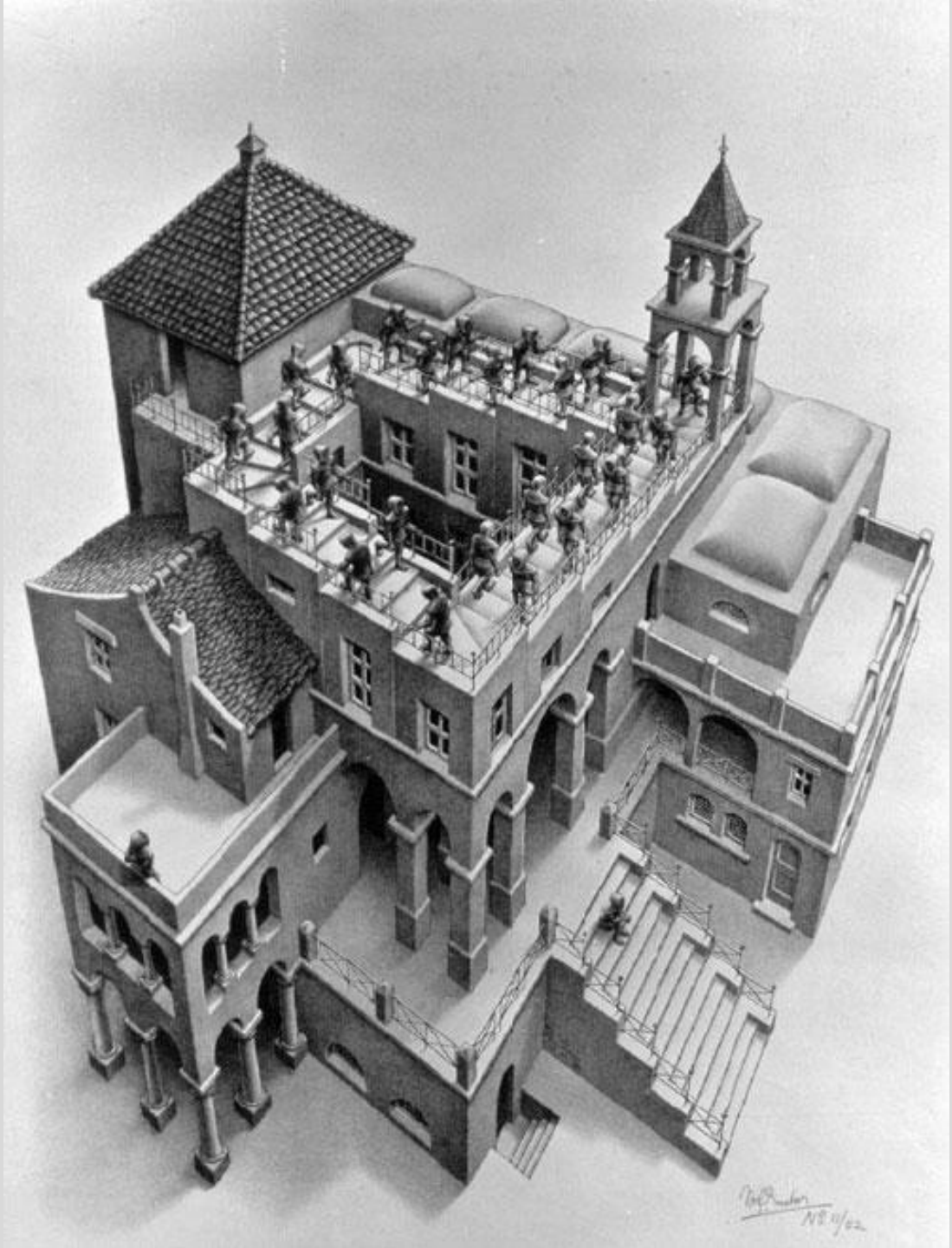
Complex object creation

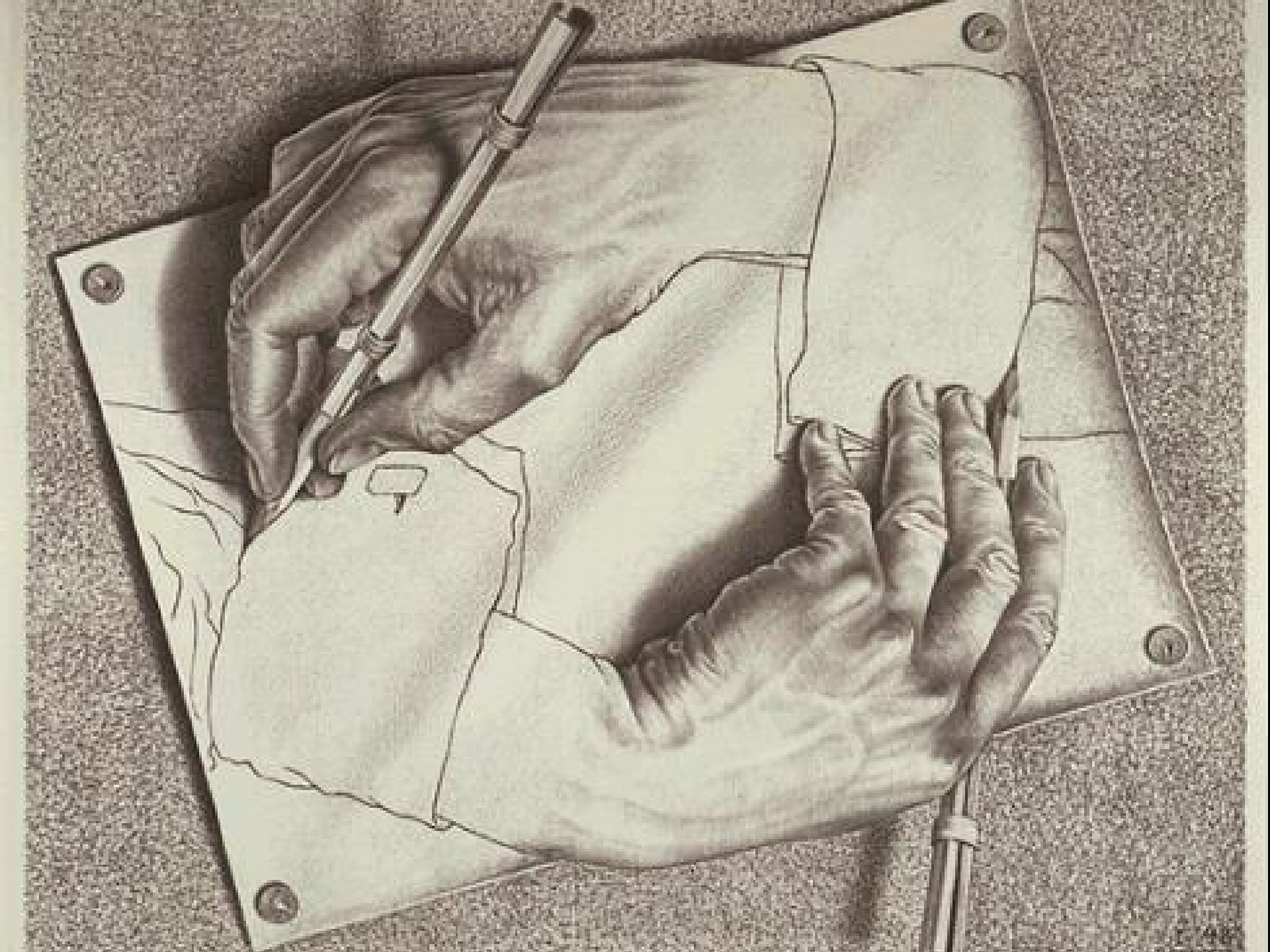
Slow test suites

Bugs

Bad Software

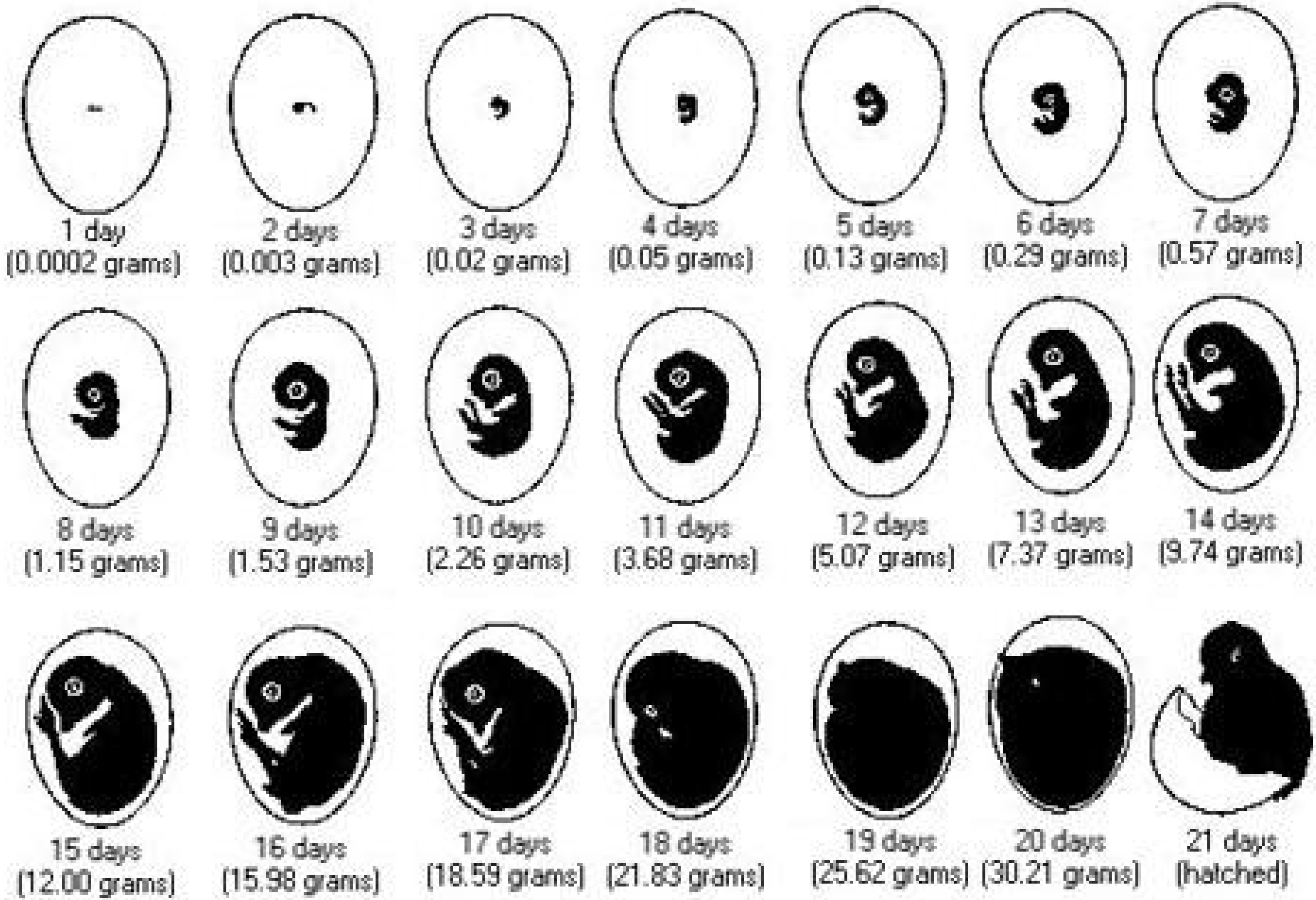
Why?





What if?

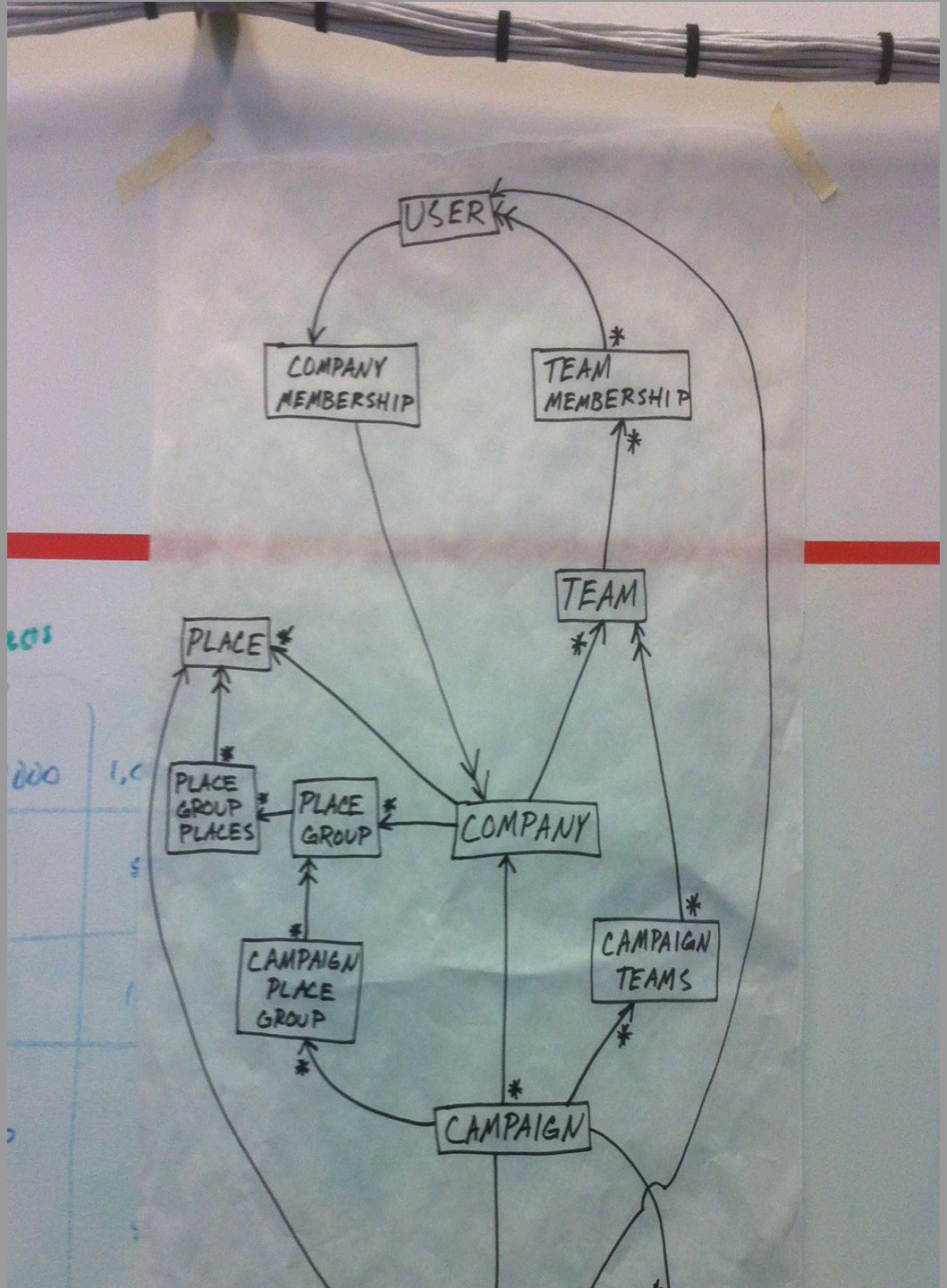

WARNING
RESTRICTED AREA
It is unlawful to enter this area without permission of the Installation Commander. (Sec 21, Internal Security Act of 1950, 50 USC 797) while on this installation, all personnel and the property under their control are subject to search.
Use of Deadly Force Authorized
VX
G
L
BZ



DAILY CHANGES IN THE WEIGHT AND FORM
OF THE DEVELOPING CHICK EMBRYO (WHITE LEGHORN)

Boulder

How?


USER
COMPANY MEMBERSHIP
TEAM MEMBERSHIP
TEAM
PLACE
PLACE GROUP PLACES
PLACE GROUP
COMPANY
CAMPAIGN PLACE GROUP
CAMPAIGN TEAMS
CAMPAIGN

Boulder

Thanks!