webmasters akademie Nürnberg GmbH

Interne DSLs in Ruby

Dipl.-Inf. (FH) Marco Emrich

Feb 2012 @ RUG

Domain Specific Language?

Mini-Sprache für spezifisches Problem

Domain Specific Language?

Mini-Sprache für spezifisches Problem

Nicht zwingend Turing-Complete

Domain Specific Language?

Mini-Sprache für spezifisches Problem

Nicht zwingend Turing-Complete

SQL

CSS

Domain Specific Language?

Mini-Sprache für spezifisches Problem

Nicht zwingend Turing-Complete

SQL

Make

CSS

Domain Specific Language?

Mini-Sprache für spezifisches Problem

Nicht zwingend Turing-Complete

XML-Configuration-Files

SQL

CSS

Make

Domain Specific Language?

Selten so **umfangreich** wie SQL oder CSS

Domain Specific Language?

Selten so **umfangreich** wie SQL oder CSS

Zoo kleiner anwendungsspezifischer DSLs

Domain Specific Language?

Selten so **umfangreich** wie SQL oder CSS

Zoo kleiner anwendungsspezifischer DSLs

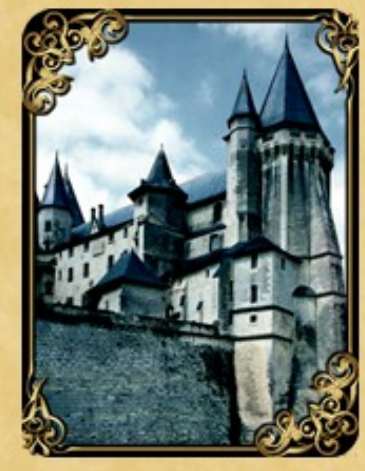
DSL
statt
API

Beispiel

Fantasy Cards RT - free Online TCG

What is Fantasy Cards RT?

Fantasy Cards RT is (probably) the first **real time online TCG** (Trading Card Game). It is a tcg, you can play online in your browser. Build your own decks and eventually even your own cards. It's, however, not finished yet. This online tcg is still an (**alpha version**).



Features

- **Online TCG**: Play it online any time
- **Browser-based**: No download required
- **Real Time**: No more waiting
- **Customizable**: Build your own **deck**
- **Free to play**: no costs at all!
- **Community Driven**: We need your ideas!

Try It

- **PLAY** the alpha version of this online tcg
- **SIGN UP** for free!
- Report bugs and discuss game ideas in the **FORUM**

Search Fantasy Cards

Google™ Custom Search

Search

Screenshots

Fantasy-Cards.net

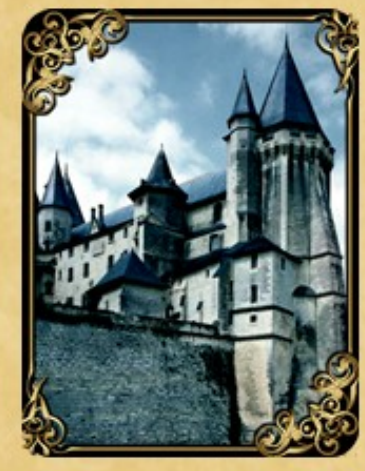
Fantasy Cards RT

Home Play Cards Forum Ranking Decks Sign Up Sign In

Fantasy Cards RT - free Online TCG

What is Fantasy Cards RT?

Fantasy Cards RT is (probably) the first **real time online TCG** (Trading Card Game). It is a tcg, you can play online in your browser. Build your own decks and eventually even your own cards. It's, however, not finished yet. This online tcg is still an (**alpha version**).



Features

- **Online TCG**: Play it online any time
- **Browser-based**: No download required
- **Real Time**: No more waiting
- **Customizable**: Build your own **deck**
- **Free to play**: no costs at all!
- **Community Driven**: We need your ideas!

Try It

- **PLAY** the alpha version of this online tcg
- **SIGN UP** for free!
- Report bugs and discuss game ideas in the **FORUM**

Search Fantasy Cards

Google™ Custom Search

Search

Screenshots

2 DSLs in Fantasy Cards

2 DSLs in Fantasy Cards

Kartenbeschreibungen

2 DSLs in Fantasy Cards

Kartenbeschreibungen

LevelScripte


Cave Dragon (2)
3
1
15
Flying
Flame Attack - 36 sec: Mass Attack to Front
Scale Growth - 36 sec, 2 Iron: +1 Armor (3 charges)
It is rumored that Cave Dragons eat iron to strengthen their scales
Dragon, Unit, Organic, Combat

Name

Attack

Armor

Health

Costs

Rules / Actions

Tags


```
dragon = Card.new('Cave Dragon', 'swamp_drake')

dragon.rules = %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i> … Cave Dragons eat iron to strengthen their scales</i>
}

dragon.add_tag(Tag.new('dragon'))
dragon.add_tag(Tag.new('unit'))
dragon.add_tag(Tag.new('organic'))
dragon.add_tag(Tag.new('combat'))

dragon.attributes = AttributeContainer.new
dragon.attributes['attack']= Attribute::Base.new(3)
dragon.attributes['armor'] = Attribute::Armor.new(1)
dragon.attributes['health']= Attribute::Health.new(1)
…[3 Zeilen gekürzt]

play = ActionBase::Play.new('play', 9)
dragon.add_action(play)

fly = ActionBase::Fly.new('fly', 9)
dragon.add_action(fly)

attack = ActionBase::Attack.new('attack', 24)
dragon.add_action(attack)

flame_attack = ActionBase::MassAttack('flame', 36)
flame_attack.target_area = Card.area_in_front_of
dragon.add_action(flame_attack)

scale_growth = ActionBase::Direct('scale_growth', 36)
scale_growth.icon = 'reinforce_armor'
scale_growth.num_charges = 3,
scale_growth.show_only_in = 'field'
scale_growth.costs = [Cost.new('iron', 2)]

def scale_growth.formula
  card[:armor] ||= 0
  card[:armor] += 1
end

dragon.add_action(scale_growth)
CardBuildersRepository.register(dragon)
```


```
dragon = Card.new('Cave Dragon', 'swamp_drake')

dragon.rules = %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i> … Cave Dragons eat iron to strengthen their scales</i>
}

dragon.add_tag(Tag.new('dragon'))
dragon.add_tag(Tag.new('unit'))
dragon.add_tag(Tag.new('organic'))
dragon.add_tag(Tag.new('combat'))

dragon.attributes = AttributeContainer.new
dragon.attributes['attack']= Attribute::Base.new(3)
dragon.attributes['armor'] = Attribute::Armor.new(1)
dragon.attributes['health']= Attribute::Health.new(1)
…[3 Zeilen gekürzt]

play = ActionBase::Play.new('play', 9)
dragon.add_action(play)

fly = ActionBase::Fly.new('fly', 9)
dragon.add_action(fly)

attack = ActionBase::Attack.new('attack', 24)
dragon.add_action(attack)

flame_attack = ActionBase::MassAttack('flame', 36)
flame_attack.target_area = Card.area_in_front_of
dragon.add_action(flame_attack)

scale_growth = ActionBase::Direct('scale_growth', 36)
scale_growth.icon = 'reinforce_armor'
scale_growth.num_charges = 3,
scale_growth.show_only_in = 'field'
scale_growth.costs = [Cost.new('iron', 2)]

def scale_growth.formula
  card[:armor] ||= 0
  card[:armor] += 1
end

dragon.add_action(scale_growth)
CardBuildersRepository.register(dragon)
```

Cave Dragon

```
Image: swamp_drake

Rules:
  Flying
  <b>Flame Attack</b> - 36 sec: Mass Attack to Front
  <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
  <i>... Cave Dragons eat iron to strengthen their scales</i>

Attributes:
  attack      3
  armor       1
  health     15
  stone_cost  2
  gold_cost   2
  power       2


Actions:
  play                9
  fly                12
  attack             24

  flame_attack       36, target_area: card.area_in_front_of

  scale_growth (direct) 36, 3 charges
    icon: reinforce_armor
    formula: armor += 1,
    show only in field,
    costs: 2 iron

Tags: dragon, unit, organic, combat
```

Cave Dragon

Image: swamp_drake

```
Rules:
  Flying
  <b>Flame Attack</b> - 36 sec: Mass Attack to Front
  <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
  <i>... Cave Dragons eat iron to strengthen their scales</i>
```

Attributes:

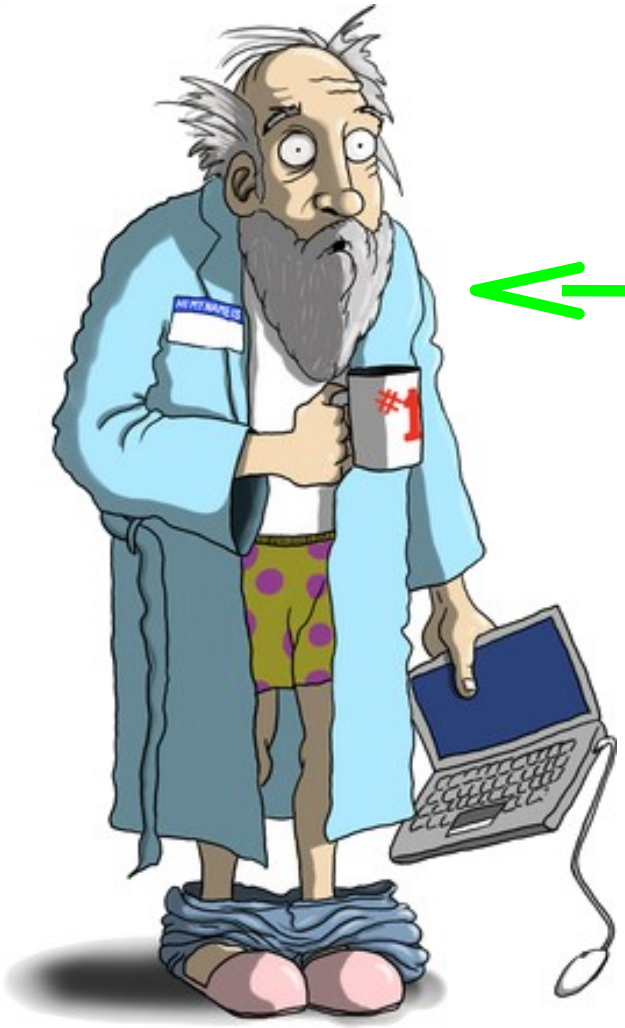
```
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
```

```
Actions:
  play                 9
  fly                 12
  attack              24

  flame_attack        36, target_area: card.area_in_front_of

  scale_growth (direct) 36, 3 charges
    icon: reinforce_armor
    formula: armor += 1,
    show only in field,
    costs: 2 iron
```

Tags: dragon, unit, organic, combat

Warum DSLs?

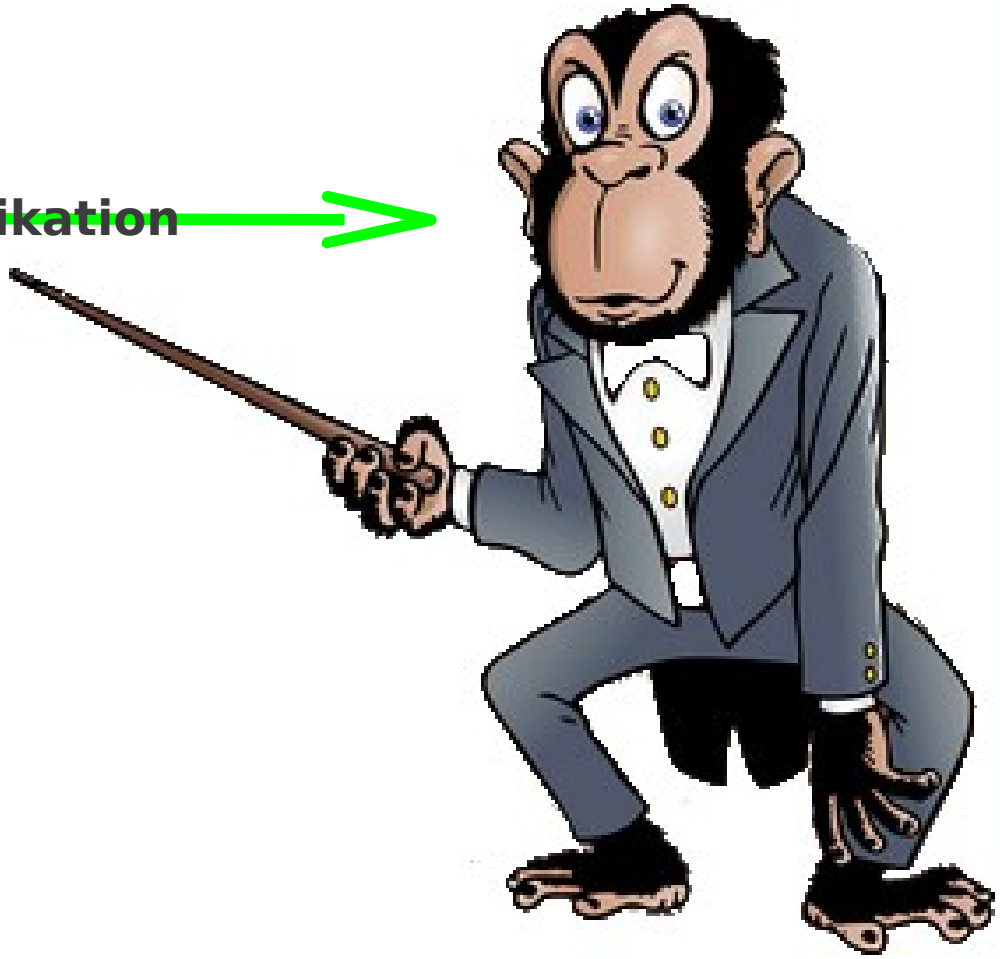
Kommunikation



Kommunikation

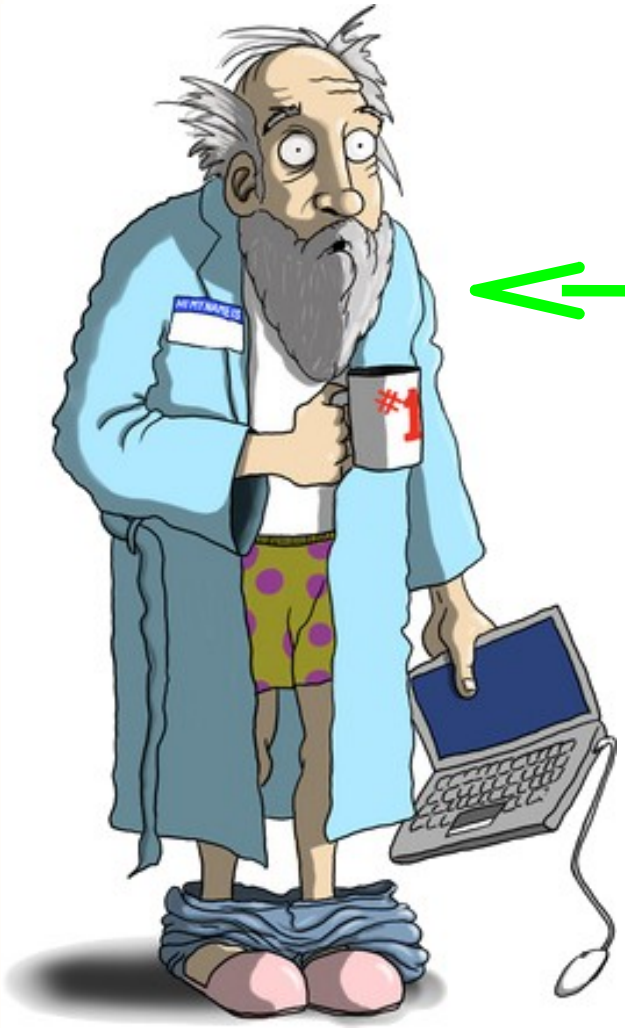
Entwickler

Kommunikation



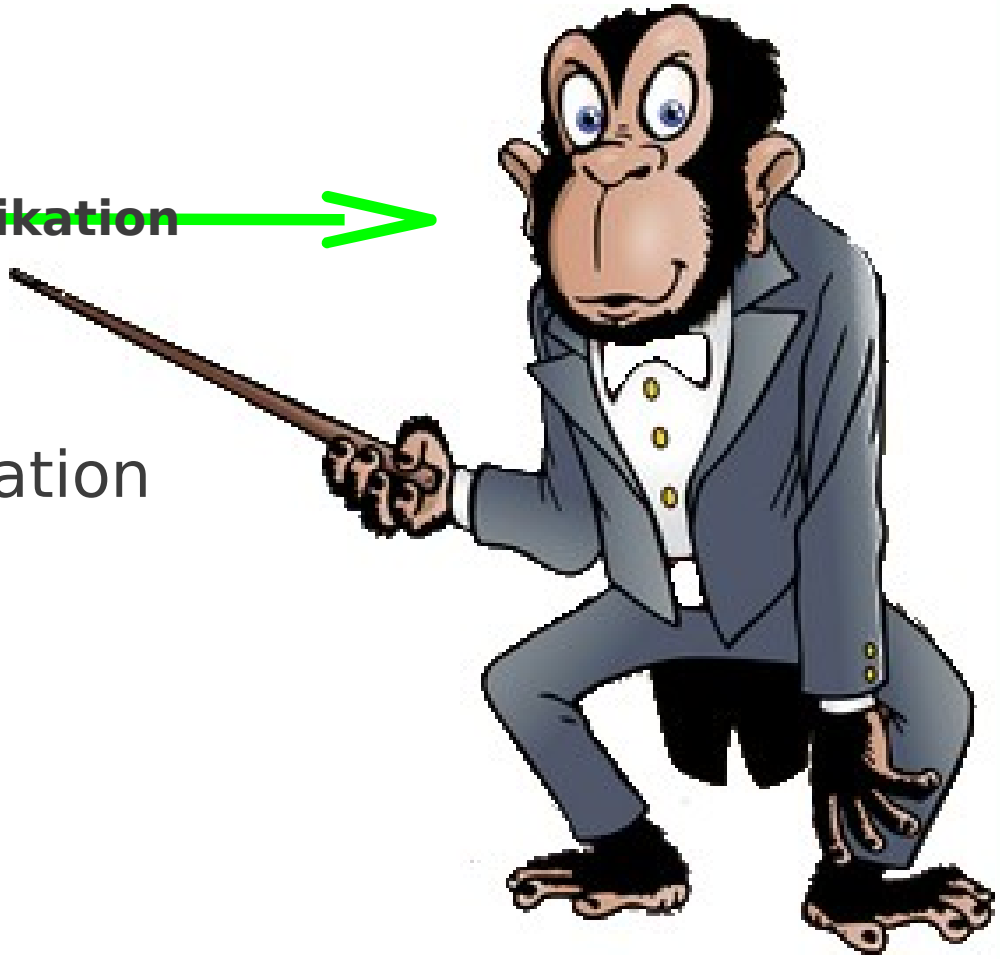
Entwickler

Fachexperte



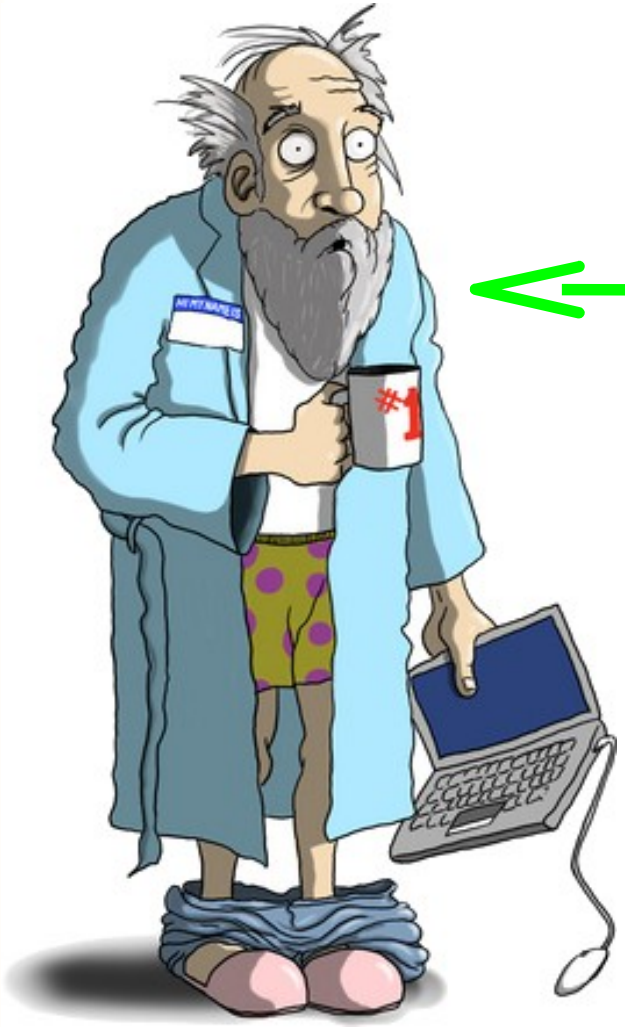
Kommunikation

Verifikation



Entwickler

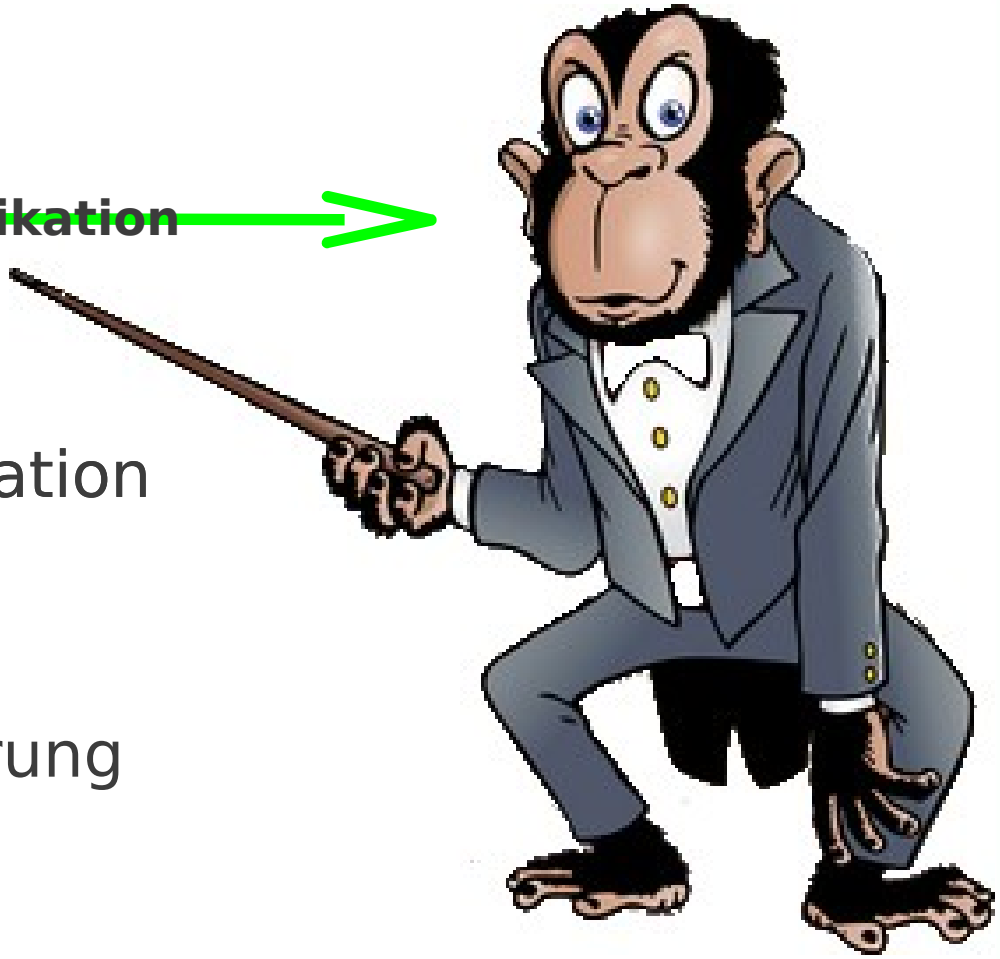
Fachexperte



Kommunikation

Verifikation

Änderung



Entwickler

Fachexperte

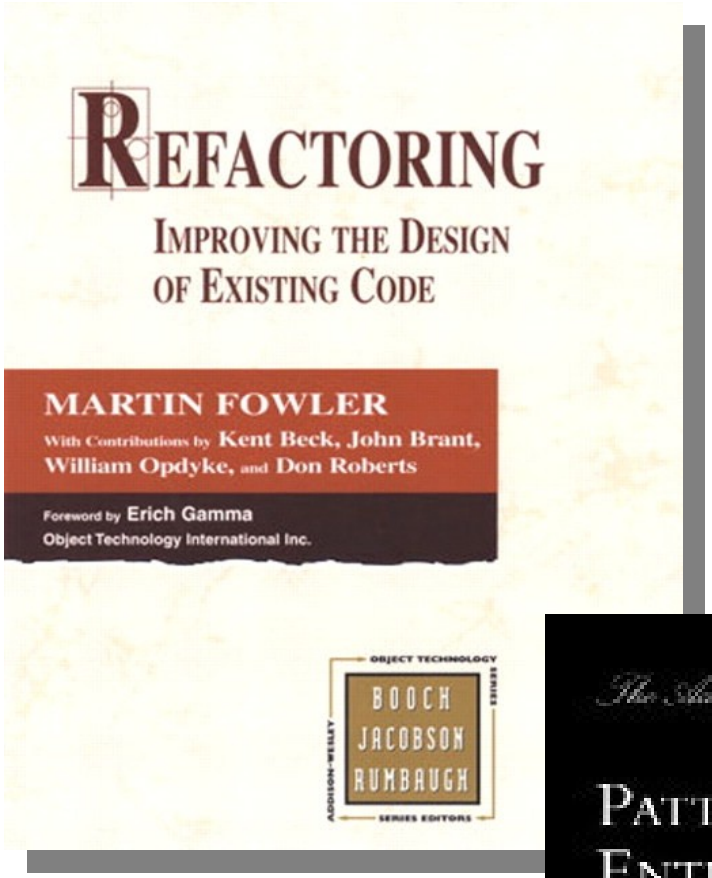
Produktivität

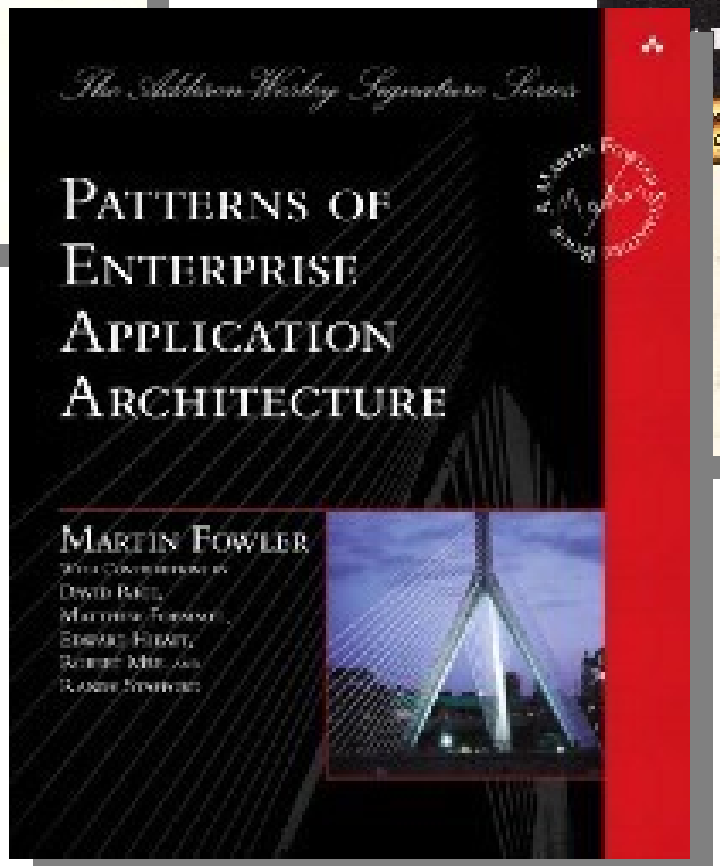
Warum Nicht?

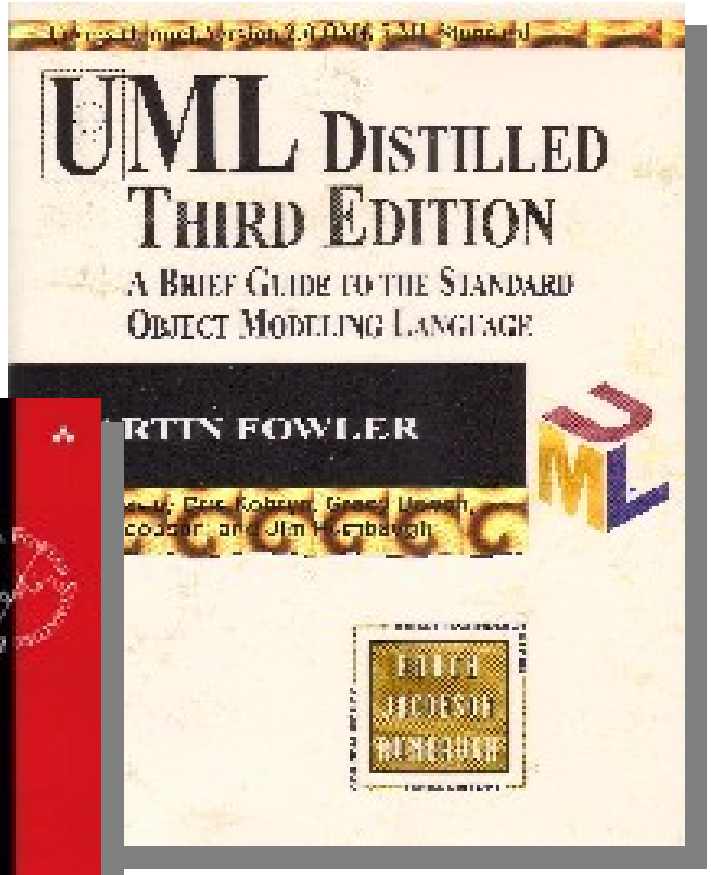
Mehraufwand

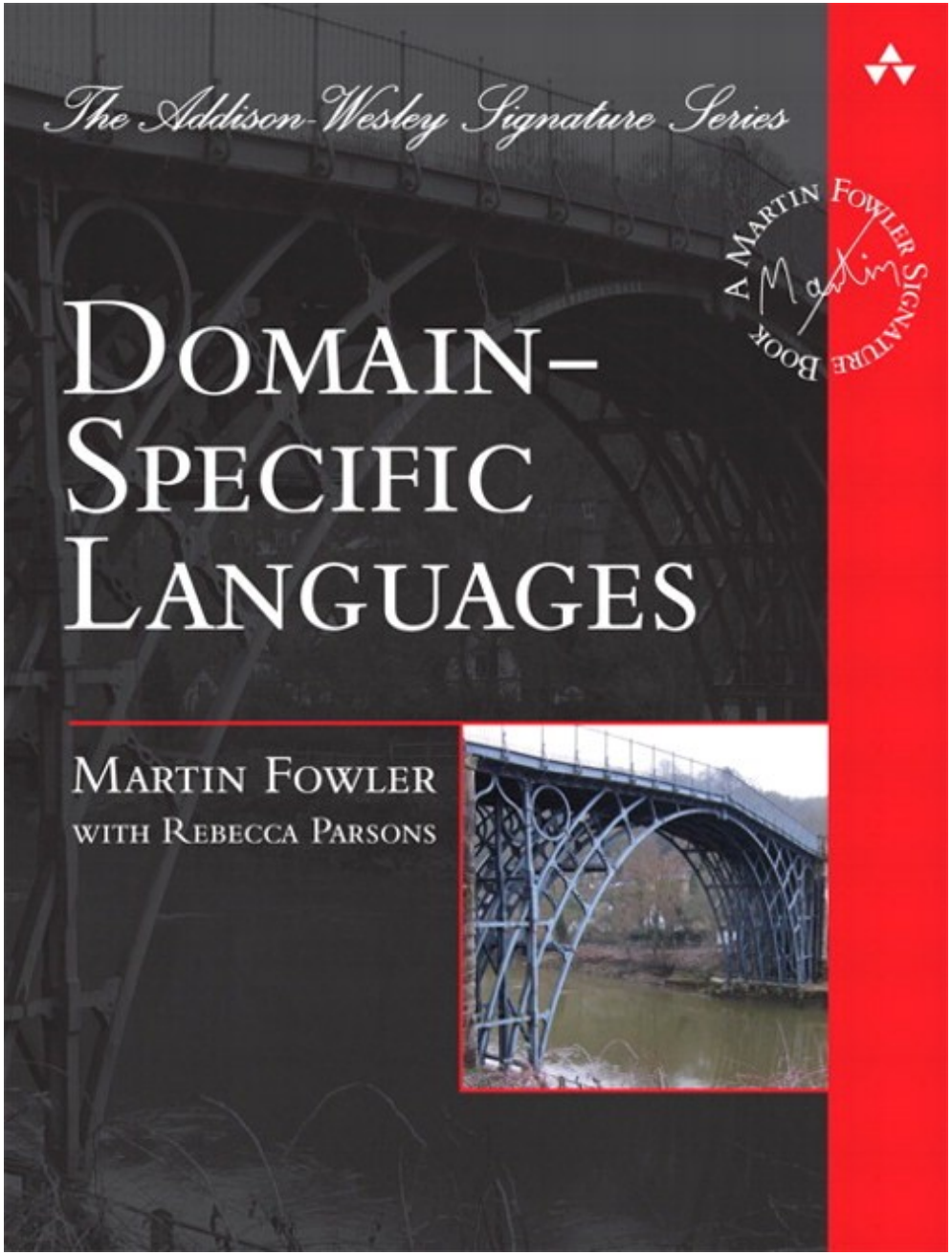
DSL Kategorien

Martin Fowler


REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
SERIES EDITORS


PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE
MARTIN FOWLER


UML DISTILLED
THIRD EDITION
A BRIEF GUIDE TO THE STANDARD OBJECT MODELING LANGUAGE
MARTIN FOWLER


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Domain-Specific Languages
Martin Fowler
with Rebecca Parsons

Interne DSL

Externe DSL

Extern

Völlig frei in der Syntax

Extern

Völlig frei in der Syntax

Parser benötigt


```
Cave Dragon

  Image: swamp_drake

  Rules:
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i>... Cave Dragons eat iron to strengthen their scales</i>

  Attributes:
    attack      3
    armor       1
    health     15
    stone_cost  2
    gold_cost   2
    power       2


  Actions:
    play                9
    fly                12
    attack             24

    flame_attack       36, target_area: card.area_in_front_of

    scale_growth (direct) 36, 3 charges
      icon: reinforce_armor
      formula: armor += 1,
      show only in field,
      costs: 2 iron

  Tags: dragon, unit, organic, combat
```

Intern

Benötigt keinen eigenen Parser

Intern

Benötigt keinen eigenen Parser

Benutzt Host-Sprache,

z.B. Ruby, LISP, Java, C#

Intern

Benötigt keinen eigenen Parser

Benutzt Host-Sprache,

z.B. Ruby, LISP, Java, C#

Gebunden an die Syntax der Host-Sprache

Intern

Benötigt keinen eigenen Parser

Benutzt Host-Sprache,

z.B. Ruby, LISP, Java, C#

Gebunden an die Syntax der Host-Sprache

Profitiert von Sprachen mit syntaktischen Freiheiten, z.B. LISP, Ruby


```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  rules %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i>... Cave Dragons eat iron to strengthen their scales</i>
  }

  attributes {
    attack       3
    armor        1
    health      15
    stone_cost   2
    gold_cost    2
    power        2
  }

  actions {
    play                9
    fly                12
    attack             24

    flame_attack       36, target_area: 'car[illegible]of'

    scale_growth :direct, 36,
      icon: 'reinforce_armor',
      num_charges: 3,
      formula: 'card[:armor] ||= 0; card[:armor] += 1',
      show_only_in: 'field',
      costs: [[:iron, 2]]
  }

  tags :dragon, :unit, :organic, :combat

end
```

Extern

- Vorteile
  - exakt an die Fachdomäne angepasst
  - keine fremden Syntaxelemente
- Nachteile
  - Parser notwendig
  - Oft aufwendig und komplex (Parserbau)

Intern

- Nachteile
  - Syntax der Hostsprache, je nach Hostsprache
- Vorteile
  - vergleichsweise einfach, kein Parserbau!
  - Nutzung von IDE / Tooling: Syntax-Highlighting, Syntax-Check, ...

Fazit

Interne DSLs sind ein guter Kompromiss

Fazit

Interne DSLs sind ein guter Kompromiss

aus **Aufwand** und **Nutzen**

Interne DSLs in Ruby

DSL-Development

DSL-Development

- Language First Design

DSL-Development

- Language First Design
- Agiles Vorgehen

Agiles Vorgehen

- Durchstich durch alle Schichten
- Nur ein Teil der Features der DSL

Obie Fernandez: DSLs für Versicherungsverträge

Obie: Klassifizierung von Ruby-DSLs

- Instantiation
- Class Macros
- Top-Level Methods

Obie: Klassifizierung von Ruby-DSLs

- Instantiation
- Class Macros
- **Top-Level Methods**

```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  rules %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i>... Cave Dragons eat iron to strengthen their scales</i>
  }

  attributes {
    attack      3
    armor       1
    health     15
    stone_cost  2
    gold_cost   2
    power       2
  }

  actions {
    play               9
    fly               12
    attack            24

    flame_attack      36, target_area: 'card.area_in_front_of'

    scale_growth :direct, 36,
      icon: 'reinforce_armor',
      num_charges: 3,
      formula: 'card[:armor] ||= 0; card[:armor] += 1',
      show_only_in: 'field',
      costs: [[:iron, 2]]
  }

  tags :dragon, :unit, :organic, :combat

end
```

```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  attributes {
    attack       3
    armor        1
    health      15
    stone_cost   2
    gold_cost    2
    power        2
  }

end
```

```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  attributes {
    attack        3
    armor         1
    health       15
    stone_cost    2
    gold_cost     2
    power         2
  }

end
```

```
def_card 'Cave Dragon' do

end
```

```
module Game

  module CardBuilderDsl

    def def_card(name, &block)
    …
    end

  end

end

include Game::CardBuilderDsl
```

```
module Game

  module CardBuilderDsl

    def def_card(name, &block)
    …
    end

  end

end

include Game::CardBuilderDsl
```

Modul/Package - Organisation

```
module Game

  module CardBuilderDsl

    def def_card(name, &block)
    …
    end

  end

end

include Game::CardBuilderDsl
```

Zieht Methodendefinition in den Kernel

```
def def_card(name, &block)
…
end
```

```
def_card 'Cave Dragon' do

end
```

```
def def_card(name, &block)
…
end
```

```
def_card('Cave Dragon')
```

Optionale Klammern

```
def def_card(name, &block)
…
end
```

```
def_card 'Cave Dragon' do

end
```

```
def def_card(name, &block)
…
end
```

```
def_card 'Cave Dragon' do

end
```

```
def def_card(name, &block)
…
end
```

```
def_card 'Cave Dragon' do
  image 'swamp_drake'

  attributes {
    attack        3
    ...
    power         2
  }

end
```

```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  attributes {
    attack        3
    armor         1
    health       15
    stone_cost    2
    gold_cost     2
    power         2
  }

end
```

&block

```
module Game
  module CardBuilderDsl
    def def_card(name, &block)
      …
    end
  end
end
include Game::CardBuilderDsl
```

```
def def_card(name, &block)
  ...
end
```

'Cave Dragon'

```
def def_card(name, &block)
  ...
end
```

'Cave Dragon'

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
def def_card(name, &block)

  ...

end
```

```
'Cave Dragon'
```

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
def def_card(name, &block)
  card = Card.new(name)
  …

end
```

```
module Game

  class Card
  …
  end

end
```

'Cave Dragon'

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
def def_card(name, &block)
  card = Card.new(name)
  …
  CardRepository.register card
end
```

```
module Game

  class Card
  …
  end

end
```

```
'Cave Dragon'
```

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
def def_card(name, &block)
  card = Card.new(name)
  block.call
  CardRepository.register card
end
```

```
module Game

  class Card
  …
  end

end
```

'Cave Dragon'

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
def def_card(name, &block)
  card = Card.new(name)
  block.bind(card).call
  CardRepository.register card
end
```

```
module Game

  class Card
  …
  end

end
```

'Cave Dragon'

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
def def_card(name, &block)
  card = Card.new(name)
  block.bind(card).call
  CardRepository.register card
end
```

```
module Game

  class Card
  …
  end

end
```

'Cave Dragon'

```
card = Card.new(name)
block.bind(card).call
```

```
class Card

  def initialize(name)
    @name = name
  end

end
```

```
card = Card.new(name)
block.bind(card).call
```

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
class Card

  def initialize(name)
    @name = name
  end

  def image(image_name)

  end

  def attributes(&block)

  end

end
```

```
card = Card.new(name)
block.bind(card).call
```

```
image 'swamp_drake'

attributes {
  attack      3
  armor       1
  health     15
  stone_cost  2
  gold_cost   2
  power       2
}
```

```
class Card

  def initialize(name)
    @name = name
  end

  def image(image_name)

  end

  def attributes(&block)

  end

end
```

```
image 'swamp_drake'

attributes {
  attack      3
  armor       1
  health     15
  stone_cost  2
  gold_cost   2
  power       2
}
```

```
card = Card.new(name)
block.bind(card).call
```

```
class Card

  def initialize(name)
    @name = name
  end

  def image(image_name)
    @image_name = image_name
  end

  def attributes(&block)

  end

end
```

```
image 'swamp_drake'

attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
card = Card.new(name)
block.bind(card).call
```

```
class Card

  def initialize(name)
    @name = name
  end

  def image(image_name)
    @image_name = image_name
  end

  def attributes(&block)
    ab = AttributesBuilder.new
    block.bind(ab).call
    @stored_attributes = ab.attributes
  end
```

```
attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```ruby
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```
attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```ruby
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```
attributes {
  attack       3
  armor        1
  health      15
  stone_cost   2
  gold_cost    2
  power        2
}
```

```
attributes {
  attack        3
  armor         1
  health       15
  stone_cost    2
  gold_cost     2
  power         2
}
```

```ruby
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```ruby
class AttributesBuilder

end
```

```
attack       3
armor        1
health      15
stone_cost   2
gold_cost    2
power        2
```

```ruby
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```ruby
class AttributesBuilder

  def initialize

  end

  def method_missing(method, value)

  end

end
```

```
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```
attack       3
armor        1
health      15
stone_cost   2
gold_cost    2
power        2
```

```
class AttributesBuilder

  def initialize

  end

  def method_missing(method, value)

  end

end
```

```
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```
attack       3
armor        1
health      15
stone_cost   2
gold_cost    2
power        2
```

```
class AttributesBuilder


  def initialize
    @attributes = {}
  end

  def method_missing(method, value)
    attr_name = method
    @attributes[attr_name] = value
  end

end
```

```
attack       3
armor        1
health      15
stone_cost   2
gold_cost    2
power        2
```

```ruby
def attributes(&block)
  ab = AttributesBuilder.new
  block.bind(ab).call
  @stored_attributes = ab.attributes
end
```

```ruby
class AttributesBuilder
  attr_reader :attributes

  def initialize
    @attributes = {}
  end

  def method_missing(method, value)
    attr_name = method
    @attributes[attr_name] = value
  end

end
```

```ruby
module Game

  module CardBuilderDsl

    def def_card(name, &block)
      card = Card.new(name)
      block.bind(card).call
      CardRepository.register card
    end

  end

end

include Game::CardBuilderDsl
```

```ruby
module Game

  class Card
    def initialize(name)
      @name = name
    end

    def image(image_name)
      @image_name = image_name
    end

    def attributes(&block)
      ab = AttributesBuilder.new
      block.bind(ab).call
      @stored_attributes = ab.attributes
    end
  end

  class AttributesBuilder
    attr_reader :attributes

    def initialize
      @attributes = {}
    end

    def method_missing(method, value)
      attr_name = method
      @attributes[attr_name] = value
    end

  end

end
```

Geringer Mehraufwand

```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  attributes {
    attack       3
    armor        1
    health      15
    stone_cost   2
    gold_cost    2
    power        2
  }

end
```

```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  rules %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i>... Cave Dragons eat iron to strengthen their scales</i>
  }

  tags :dragon, :unit, :organic, :combat

  attributes {
    attack      3
    armor       1
    health     15
    stone_cost  2
    gold_cost   2
    power       2
  }

  actions {
    play               9
    fly               12
    attack            24

    flame_attack      36, target_area: 'card.area_in_front_of'

    scale_growth :direct, 36,
      icon: 'reinforce_armor',
      num_charges: 3,
      formula: 'card[:armor] ||= 0; card[:armor] += 1',
      show_only_in: 'field',
      costs: [[:iron, 2]]
  }

end
```


```
def_card 'Cave Dragon' do

  image 'swamp_drake'

  rules %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i>... Cave Dragons eat iron to strengthen their scales</i>
  }

  tags :dragon, :unit, :organic, :combat

  attributes {
    attack       3
    armor        1
    health      15
    stone_cost   2
    gold_cost    2
    power        2
  }

  actions {
    play                9
    fly                12
    attack             24

    flame_attack       36, target_area: 'card.area_in_front_of'

    scale_growth :direct, 36,
      icon: 'reinforce_armor',
      num_charges: 3,
      formula: 'card[:armor] ||= 0; card[:armor] += 1',
      show_only_in: 'field',
      costs: [[:iron, 2]]
  }

end
```

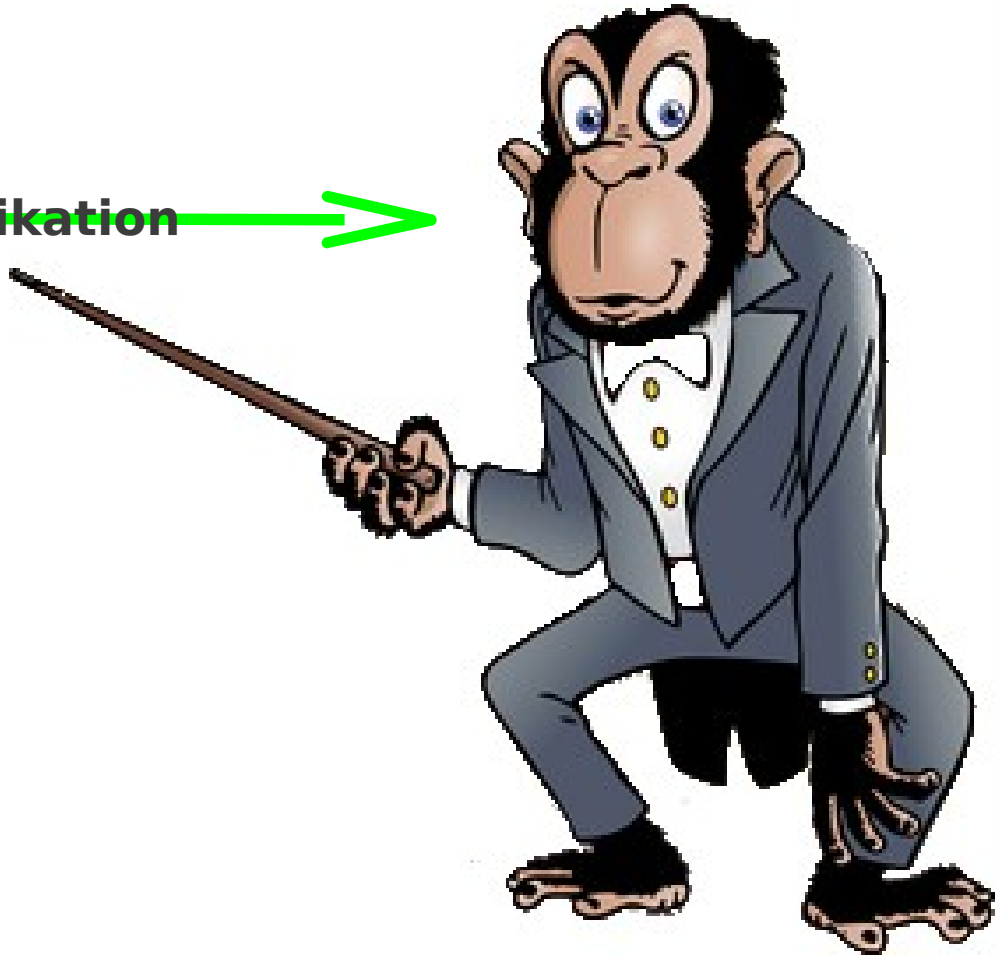
Ruby-DSLs leiden nur geringfügig unter der Host-Syntax


```
dragon = Card.new('Cave Dragon', 'swamp_drake')

dragon.rules = %{
    Flying
    <b>Flame Attack</b> - 36 sec: Mass Attack to Front
    <b>Scale Growth</b> - 36 sec, 2 Iron: +1 Armor (3 charges)
    <i> … Cave Dragons eat iron to strengthen their scales</i>
}

dragon.add_tag(Tag.new('dragon'))
dragon.add_tag(Tag.new('unit'))
dragon.add_tag(Tag.new('organic'))
dragon.add_tag(Tag.new('combat'))

dragon.attributes = AttributeContainer.new
dragon.attributes['attack']= Attribute::Base.new(3)
dragon.attributes['armor'] = Attribute::Armor.new(1)
dragon.attributes['health']= Attribute::Health.new(1)
…[3 Zeilen gekürzt]

play = ActionBase::Play.new('play', 9)
dragon.add_action(play)

fly = ActionBase::Fly.new('fly', 9)
dragon.add_action(fly)

attack = ActionBase::Attack.new('attack', 24)
dragon.add_action(attack)

flame_attack = ActionBase::MassAttack('flame', 36)
flame_attack.target_area = Card.area_in_front_of
dragon.add_action(flame_attack)

scale_growth = ActionBase::Direct('scale_growth', 36)
scale_growth.icon = 'reinforce_armor'
scale_growth.num_charges = 3,
scale_growth.show_only_in = 'field'
scale_growth.costs = [Cost.new('iron', 2)]

def scale_growth.formula
  card[:armor] ||= 0
  card[:armor] += 1
end

dragon.add_action(scale_growth)
CardBuildersRepository.register(dragon)
```

DSL-Code deutlich lesbarer

Produktivität

Kommunikation



Entwickler

Fachexperte

Lust auf eigene Experimente?

Weiterführende Links

http://martinfowler.com/bliki/DomainSpecificLanguage.html

http://martinfowler.com/bliki/DslQandA.html

http://www.infoq.com/presentations/agile-dsl-development-in-ruby

http://blog.jayfields.com/2008/02/implementing-internal-dsl-in-ruby.html

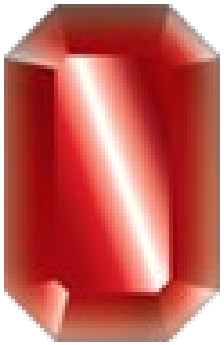
https://github.com/bobsh/puppet-rubydsl-examples

http://www.artima.com/weblogs/viewpost.jsp?thread=251945

http://www.khelll.com/blog/ruby/ruby-and-internal-dsls/

http://www.infoq.com/news/2007/06/dsl-or-not

JRuby



IronRuby

Seminare & Consulting

www.webmasters-akademie.de

m.emrich@webmasters.de

https://www.xing.com/profile/Marco_Emrich3

http://twitter.com/marcoemrich