

Nicolai Constantin Reuschling

Lyrik, Prosa, Ruby & Rails

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit Ruby sehr schnell und außerdem sehr einfach definieren.
- » Ruby on Rails macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind Rails-Applikationen *wartungsfreundlich*.

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit Ruby sehr schnell und außerdem sehr einfach definieren.
- » Ruby on Rails macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind Rails-Applikationen *wartungsfreundlich*.

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit Ruby sehr schnell und außerdem sehr einfach definieren.
- » Ruby on Rails macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind Rails-Applikationen *wartungsfreundlich*.

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit Ruby sehr schnell und außerdem sehr einfach definieren.
- » Ruby on Rails macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind Rails-Applikationen *wartungsfreundlich*.

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit **Ruby** sehr schnell und außerdem sehr einfach definieren.
- » Ruby on Rails macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind Rails-Applikationen *wartungsfreundlich*.

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit **Ruby** sehr schnell und außerdem sehr einfach definieren.
- » **Ruby on Rails** macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind Rails-Applikationen *wartungsfreundlich*.

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit **Ruby** sehr schnell und außerdem sehr einfach definieren.
- » **Ruby on Rails** macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind **Rails**-Applikationen *wartungsfreundlich*.

Relevanz

– Was bedeutet eigentlich Wartung bei Web-Applikationen? –

Lange Entwicklungszeiten
(Monate/Jahre)



Kurze Entwicklungsintervalle
(Wochen/Tage/**Stunden!**)

Relativ gute Planungssicherheit



Anforderungsdynamik

„Shipping“
(CD & DVD – Trend geht zu Download)



„Deployment“
(im besten Fall: **sofort!**)

Hohe Kundenbindung
(aus Gründen der Kompatibilität)



Niedrigere Kundenbindung

Großer Anteil Standardsoftware
(Microsoft, Adobe, SAP usw.)



Meist Individualsoftware
(„Software als Service“; mitunter
sogar schon „serviceorientiert“)

Vorgehensmodell: agil ✓

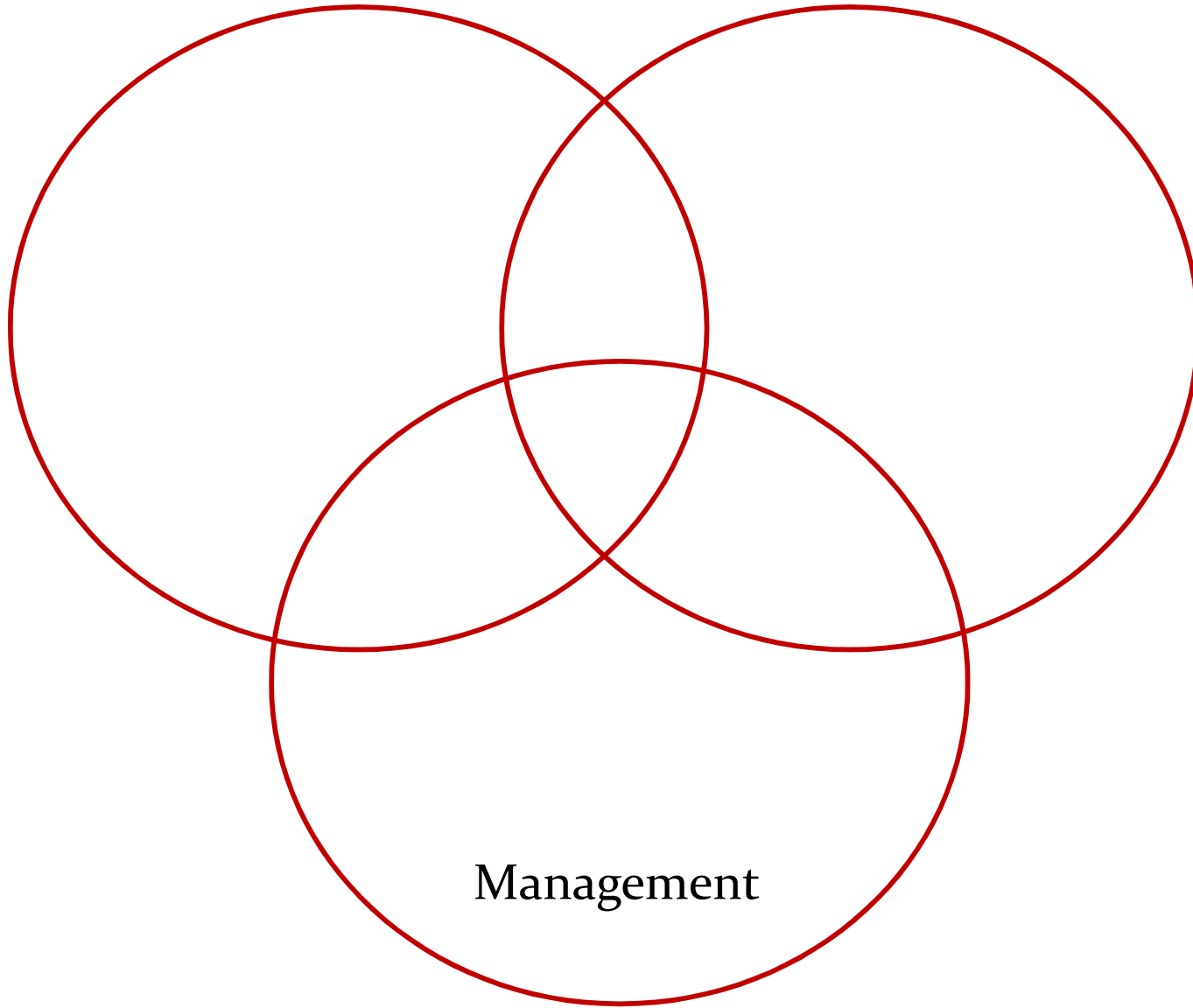
Brauchbare Werkzeuge: ✗

Problematik

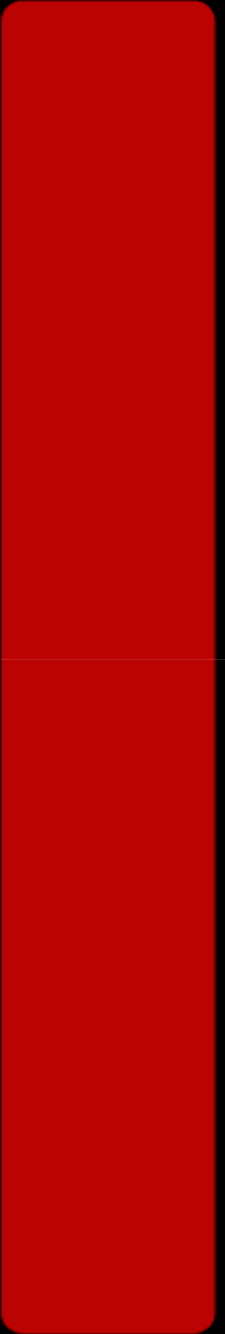
– Wieso ist Software eigentlich nicht „per se“ wartbar? –

Computer Programming

Application Engineering



Management





„Application Engineering“

„Computer Programming“



» **Semantik**

- » Was? Warum?
- » eher lösungsorientiert
- » abstrakt/deklarativ
- » leicht verständlich
- » flexibler

» **Syntax**

- » Wie?
- » eher problemorientiert
- » konkret/algorithmisch
- » schwer verständlich
- » festgelegt

» **Semantik**

» **Was? Warum?**

» eher lösungsorientiert

» abstrakt/deklarativ

» leicht verständlich

» flexibler

» **Syntax**

» **Wie?**

» eher problemorientiert

» konkret/algorithmisch

» schwer verständlich

» festgelegt

- » Semantik
- » Was? Warum?
- » eher lösungsorientiert
- » abstrakt/deklarativ
- » leicht verständlich
- » flexibler

- » Syntax
- » Wie?
- » eher problemorientiert
- » konkret/algorithmisch
- » schwer verständlich
- » festgelegt

- » Semantik
- » Was? Warum?
- » eher lösungsorientiert
- » abstrakt/deklarativ
- » leicht verständlich
- » flexibler

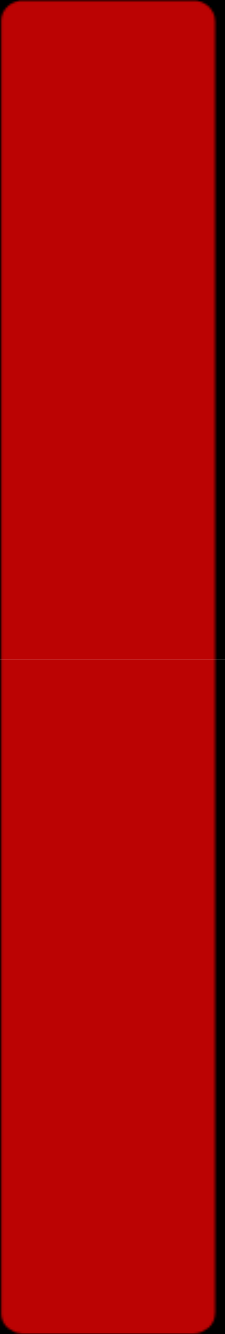
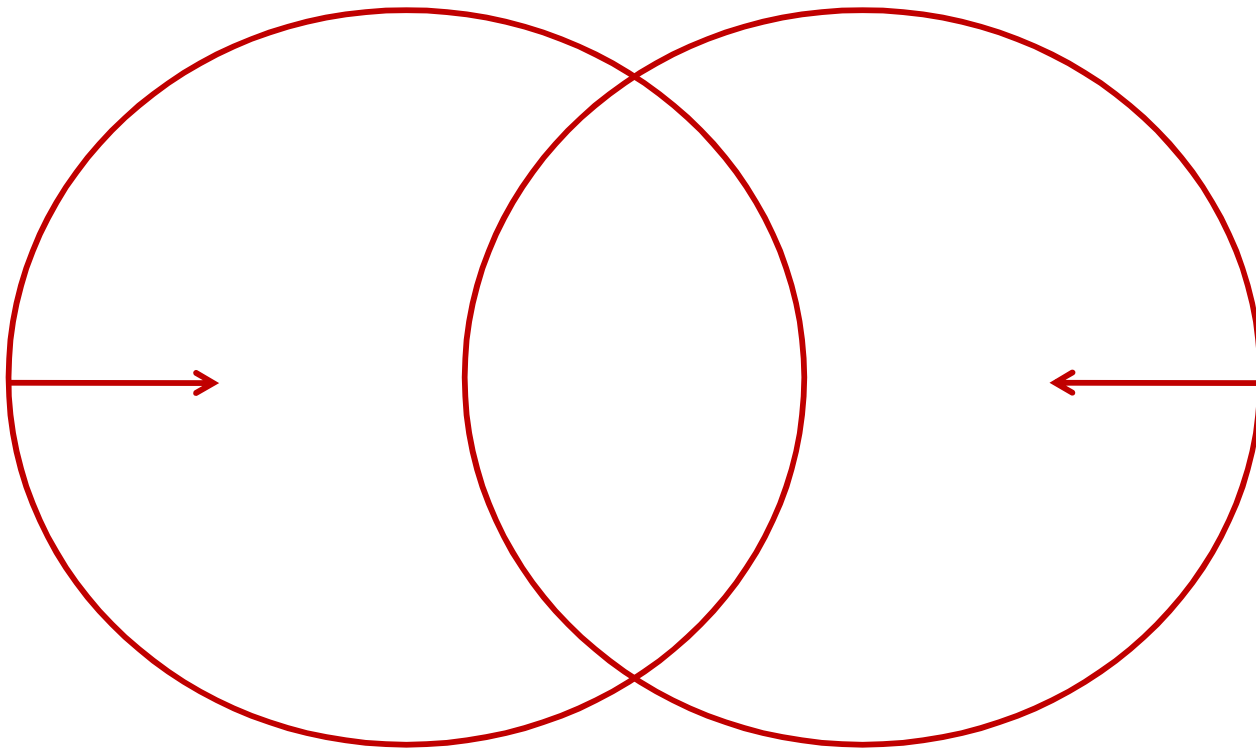
- » Syntax
- » Wie?
- » eher problemorientiert
- » konkret/algorithmisch
- » schwer verständlich
- » festgelegt

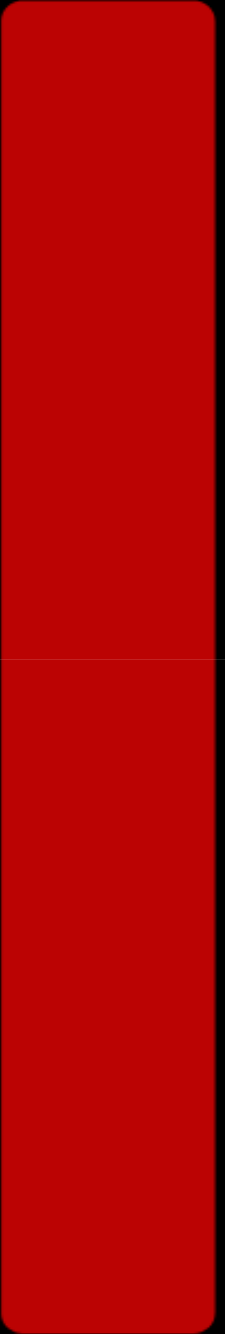
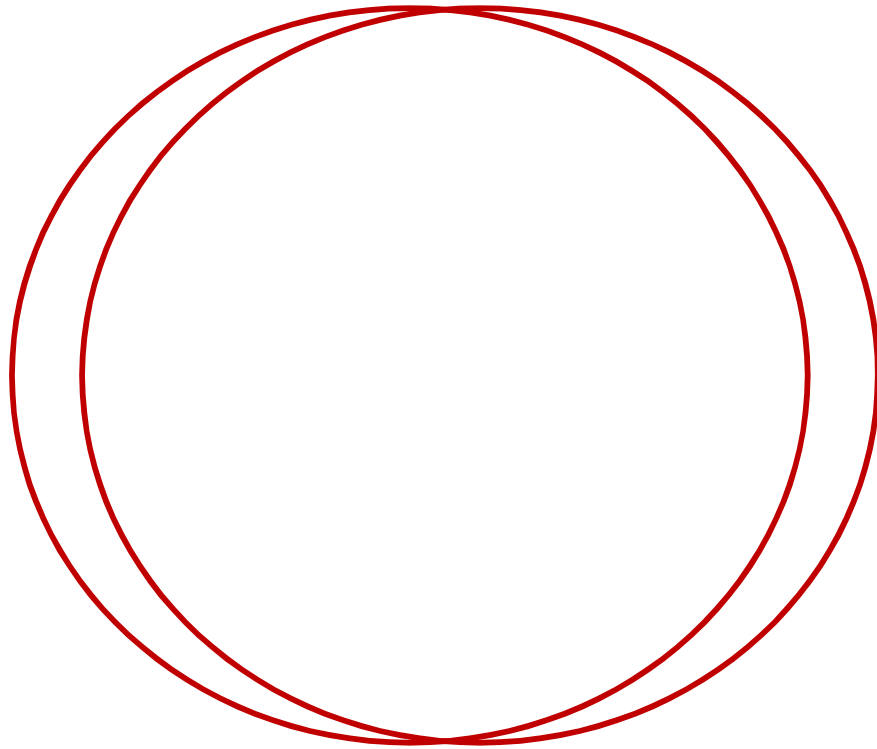
- » Semantik
- » Was? Warum?
- » eher lösungsorientiert
- » abstrakt/deklarativ
- » leicht verständlich
- » flexibler

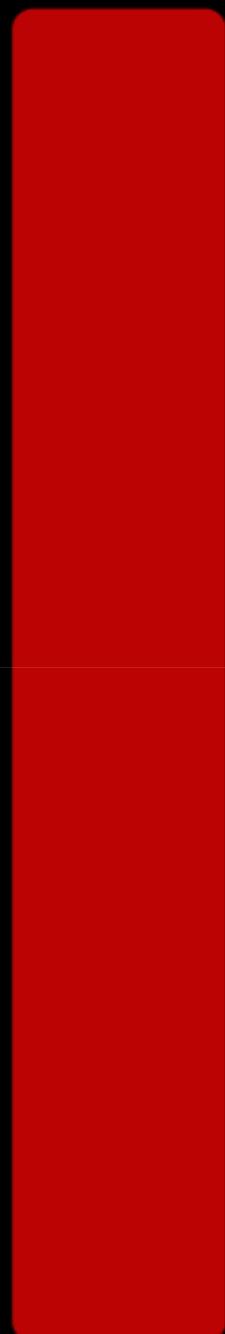
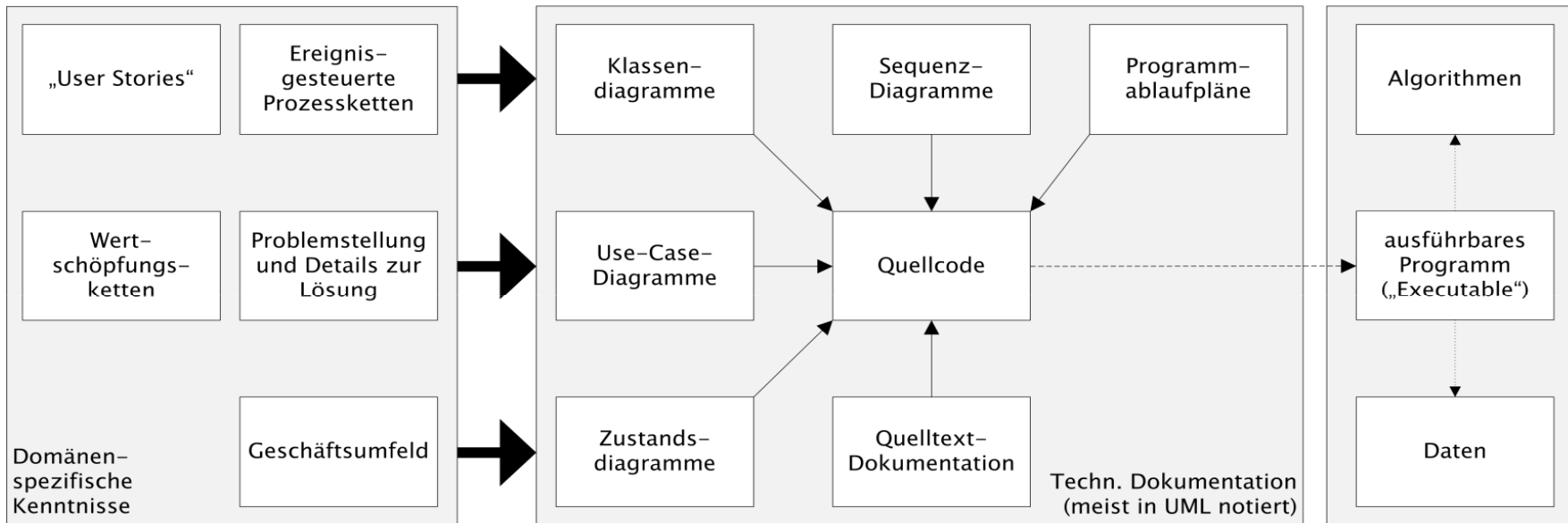
- » Syntax
- » Wie?
- » eher problemorientiert
- » konkret/algorithmisch
- » schwer verständlich
- » festgelegt

- » Semantik
- » Was? Warum?
- » eher lösungsorientiert
- » abstrakt/deklarativ
- » leicht verständlich
- » flexibler

- » Syntax
- » Wie?
- » eher problemorientiert
- » konkret/algorithmisch
- » schwer verständlich
- » festgelegt







Zu Hilfe kommen uns:

Domänenspezifische Sprachen
(Domain Specific Languages, kurz: DSL)

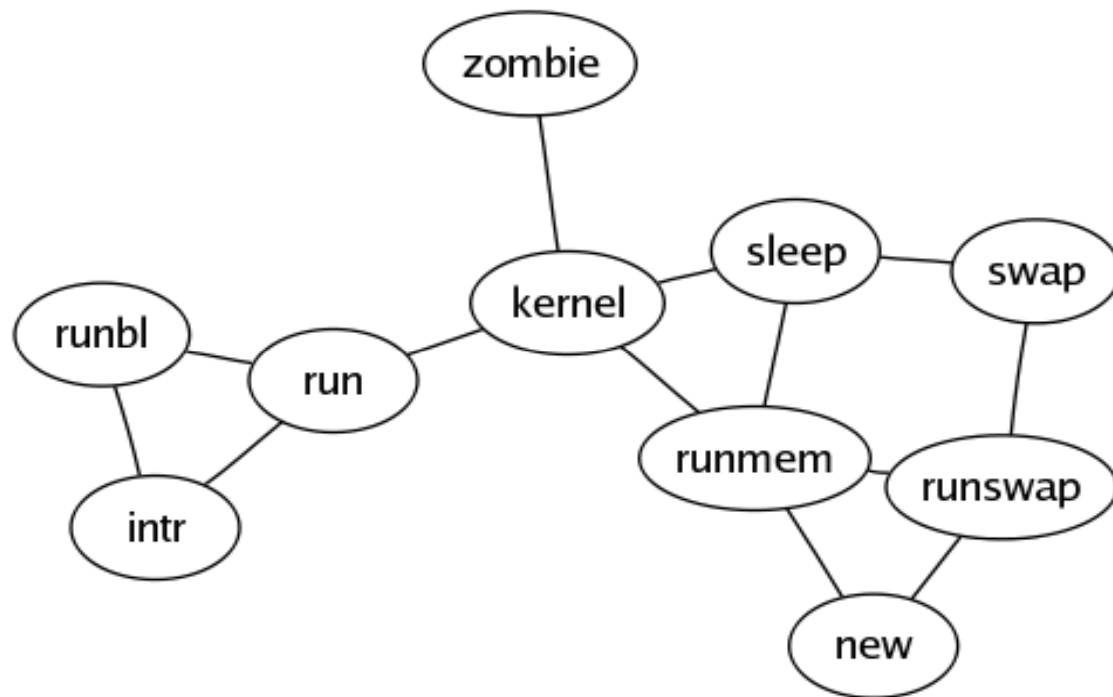
„Ein domänenspezifische Sprache
löst eine Aufgabe
mit und in der Sprache des Problems.“

DSL ◀▶ UML

Graphviz Library

(hier: Prozesszustände im Unix-Kernel)

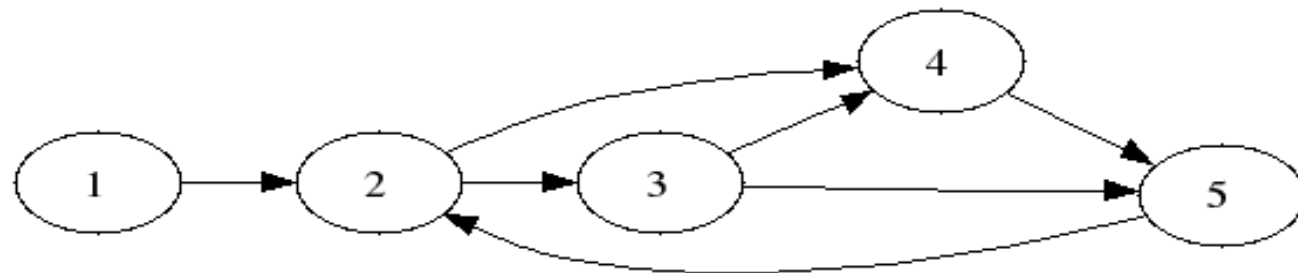
```
graph G {
    run -- intr;
    intr -- runbl;
    runbl -- run;
    run -- kernel;
    kernel -- zombie;
    kernel -- sleep;
    kernel -- runmem;
    sleep -- swap;
    swap -- runswap;
    runswap -- new;
    runswap -- runmem;
    new -- runmem;
    sleep -- runmem;
}
```



Ein anderes Beispiel

(diesmal: gerichteter Graph)

```
digraph G {  
  rankdir=LR  
  1 -> 2;  
  2 -> 3;  
  2 -> 4;  
  3 -> 4;  
  3 -> 5;  
  4 -> 5;  
  5 -> 2;  
}
```



Ruby

– Das „Juwel“ unter der Programmiersprachen –

Ruby...

- » lässt sich *vollständig* objektorientiert programmieren
- » ist ausnahmslos dynamisch erweiterbar
- » ist besonders gut lesbar
- » erleichtert Programmierern die Arbeit
- » enthält viel „Syntactic Sugar“
- » ist optimiert auf die Produktivität der Programmierer, *nicht* auf Laufzeit-Performanz
- » eignet sich hervorragend für die Implementierung von domänenspezifischen Sprachen

```
puts("Schau mal, Mama, ohne Füße!");
```

```
puts("Schau mal, Mama, ohne Hände!");
```

```
puts "Und jetscht: ohne Tschähne!"
```



```
// bad
if(number >= 23 && number <= 42) {
    doSomething();
}
```

```
# good
if (23..42).include? number then
    do_something
end
```

```
// bad
if(number >= 23 && number <= 42) {
    doSomething();
}
```

```
# good
if (23..42).include? number then
    do_something
end
```

```
// bad
if(!currentUser.isExternal()) {
  doSomething();
}
```

```
# good
do_something unless current_user.is_external?
```

```
// bad
if(!currentUser.isExternal()) {
  doSomething();
}
```

```
# good
do_something unless current_user.is_external?
```

```
// bad
temp = object1;
object1 = object2;
object2 = temp;
```

```
# good
object1, object2 = object2, object1
```

```
// bad
temp = object1;
object1 = object2;
object2 = temp;
```

```
# good
object1, object2 = object2, object1
```

```
Mouse mouse = new Mouse ();
```

```
Cat cat = new Cat ();
```

```
Dog dog = new Dog ();
```

```
Vector pets = new Vector ();
```

```
pets.add(mouse);
```

```
pets.add(cat);
```

```
pets.add(dog);
```

```
same_mouse = pets.elementAt(0);
```

```
same_cat    = pets.elementAt(1);
```

```
same_dog    = pets.elementAt(2);
```

```
Mouse mouse = new Mouse ();  
Cat cat = new Cat ();  
Dog dog = new Dog ();
```

```
Vector pets = new Vector ();  
pets.add(mouse);  
pets.add(cat);  
pets.add(dog);
```

```
Mouse same_mouse = (Mouse) pets.elementAt (0);  
Cat same_cat = (Cat) pets.elementAt (1);  
Dog same_dog = (Dog) pets.elementAt (2);
```

```
# very good
3.days.from_now
17.weeks.ago
(4.years + 1.days).from_now
```

```
600.megabyte + 600.megabyte > 1.gigabyte
```

```
5.kilometer.from(:current_location)
```

```
cars.reject {|car| car.broken?}
```

```
numbers.select {|number| number.prime?}
```

```
xml.instruct!  
xml.rss "version" => "2.0",  
      "xmlns:dc" => "http://purl.org/dc/elements/1.1/" do  
  xml.channel do  
    xml.title 'News'  
    xml.link 'http://domain-name.tld/newsfeed.rss'  
    xml.pubDate News.publish_date  
    xml.description 'News Feed'  
    news.each do |entry|  
      xml.item do  
        xml.title entry.title  
        xml.link entry.link  
        xml.description entry.description  
        xml.pubDate entry.publish_date  
        xml.guid entry.guid  
        xml.author entry.author_name  
      end  
    end  
  end  
end  
end  
end
```

```
<?xml version="1.0" encoding="UTF-8"?>
  <rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
    <channel>
      <title>News</title>
      <link>http://domain-name.tld/newsfeed.rss</link>
      <pubDate>Wed, 28 Mar 2007 16:03:50 GMT</pubDate>
      <description>News Feed</description>
      <item>
        <title>Child hostages freed in Manila</title>
        <link>http://domain-name.tld/news/entry/4711</link>
        <description>A daycare operator -- armed with grenades and a gun
-- released more than 30 preschool children and their teachers
Wednesday after holding them hostage on a bus for more than eight
hours.</description>
        <pubDate>Wed, 28 Mar 2007 16:00:00 GMT</pubDate>
        <guid>http://domain-name.tld/news/entry/4711</guid>
        <author>Max Mustermann</author>
      </item>
    </channel>
  </rss>
```

Ruby on Rails

- DSL 1337 -

Eine Bachelorarbeit...

- » hat einen Autor
- » hat einen Titel
- » hat einen Untertitel
- » hat eine Abgabetermin
- » hat verschiedenen Seiten
- » bekommt eine Note zugeordnet

Eine Bachelorarbeit...

- » hat einen Autor (mit Vor- und Zuname)
- » hat einen einmaligen Titel
- » hat einen optionalen Untertitel
- » hat eine Abgabetermin (drei Monate)
- » hat verschiedene (geordneten) Seiten
- » bekommt eine Note zugeordnet (1,0–5,0)

Author

requires first_name, last_name
belongs to a bachelor thesis

Page

belongs to a bachelor thesis
requires content

Bachelor Thesis

has one author

has many pages

unique title

due_date = 3 months

grade = [1.0, 1.3, 1.7...]

```
class BachelorThesis < ActiveRecord::Base
  VALID_GRADES = ['1.0', '1.3', '1.7',
                 '2.0', '2.3', '2.7',
                 '3.0', '3.3', '3.7',
                 '4.0', '5.0', ''].freeze

  has_one :author
  has_many :pages

  validates_inclusion_of :grade, :in => VALID_GRADES
  validates_presence_of :title, :due_date
  validates_uniqueness_of :title

  def before_validation_on_create
    self.due_date ||= 3.months.from_now
  end
end
```

```
class Author < ActiveRecord::Base
  belongs_to :bachelor_thesis
  validates_presence_of :first_name, :last_name
end
```

```
class Page < ActiveRecord::Base
  belongs_to :bachelor_thesis

  acts_as_list :scope => :bachelor_thesis

  validates_presence_of :content
  validates_presence_of :bachelor_thesis_id
end
```

```
class BachelorThesis < ActiveRecord::Base
```

```
  VALID_GRADES = ['1.0', '1.3', '1.7',  
                 '2.0', '2.3', '2.7',  
                 '3.0', '3.3', '3.7',  
                 '4.0', '5.0', ''].freeze
```

```
  has_one :author
```

```
  has_many :pages
```

```
  validates_inclusion_of :grade, :in => VALID_GRADES
```

```
  validates_presence_of :title, :due_date
```

```
  validates_uniqueness_of :title
```

```
  def before_validation_on_create
```

```
    self.due_date ||= 3.months.from_now
```

```
  end
```

```
end
```

```
ActiveRecord::Schema.define(:version => 5) do

  create_table "authors", :force => true do |t|
    t.column "first_name",      :string
    t.column "last_name",       :string
    t.column "bachelor_thesis_id", :integer
  end

  create_table "bachelor_theses", :force => true do |t|
    t.column "title",          :string
    t.column "subtitle",       :string
    t.column "grade",          :string
    t.column "due_date",       :datetime
  end

  create_table "pages", :force => true do |t|
    t.column "content",        :text
    t.column "bachelor_thesis_id", :integer
    t.column "position",       :integer
  end

end
```

Resümee

- Rails has_much :potential -

„...in a nutshell“

- » Web-Applikationen sind wesentlich wartungsintensiver als klassische Desktop- und auch Server-Applikationen.
- » Die Lesbarkeit des Quelltexts beeinflusst die Wartbarkeit der gesamten Anwendung.
- » Domänenspezifische Sprachen sind besonders gut lesbar.
- » Solche Sprachen lassen sich mit **Ruby** sehr schnell und außerdem sehr einfach definieren.
- » **Ruby on Rails** macht für die Modellierung intensiven Gebrauch davon.
- » Folglich sind **Rails**-Applikationen *wartungsfreundlich*.

“I conclude that there are two ways
of constructing a software design:
One way is to make it so simple
that there are obviously no deficiencies
and the other way is to make it so complicated
that there are no obvious deficiencies.”

—Sir Charles Antony Richard Hoare

<http://nicolai.reuschling.name/bachelorthesis/>



Fragen?

Fragen!

Ende

- Leider ohne „r“ -