Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

The 22th International Conference on Web Engineering (ICWE-2022)
Juli 5-8, 2022, Bari, Italy

Tutorial: About Lightweight Code Generation

Andreas Schmidt

andreas.schmidt@kit.edu
Institute for Automation and
Applied Informatics
Karlsruhe Institute of Technologie
Germany

andreas.schmidt@h-ka.de
Faculty of Computer Science and Business
Information Systems
University of Applied Sciences Karlsruhe
Germany

v20220708.1452

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Time Schedule

Time	July 8
09:00 - 10:30	T1: ROOM 4 (Aula 4) T2: ROOM 6 (Aula 6) T4: ROOM 8 (Aula 8) W3: ROOM 10 (Aula 10) W5: ROOM 12 (Aula 12)
10:30 - 11:00	**BREAK**
11:00 ~~11:30~~ - 12:30	T1: ROOM 4 (Aula 4) T2: ROOM 6 (Aula 6) T4: ROOM 8 (Aula 8) W3: ROOM 10 (Aula 10) W5: ROOM 12 (Aula 12)

Purpose

- Present the principal architecture and functioning of a code generator
- Enable the participants to
  - ... use existing code generators
  - ... build generators on their own

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

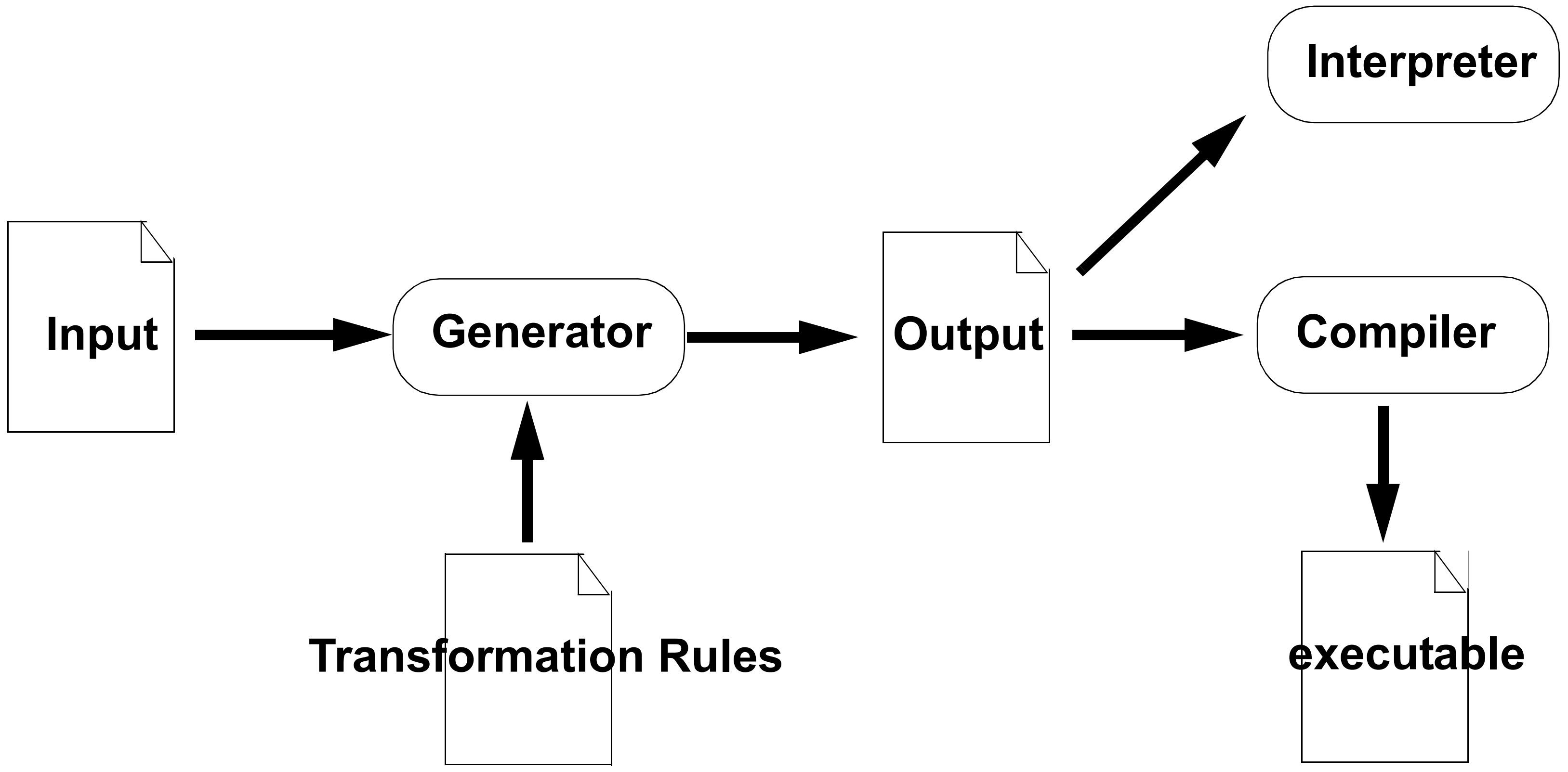
Outline

- Introduction and Motivation
- Regular Expressions
- Overview of different generator technologies
- Practical examples and exercices
- Summary and next steps

Hochschule Karlsruhe
University of
Applied Sciences

What is a Software Generator

- Principle



Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

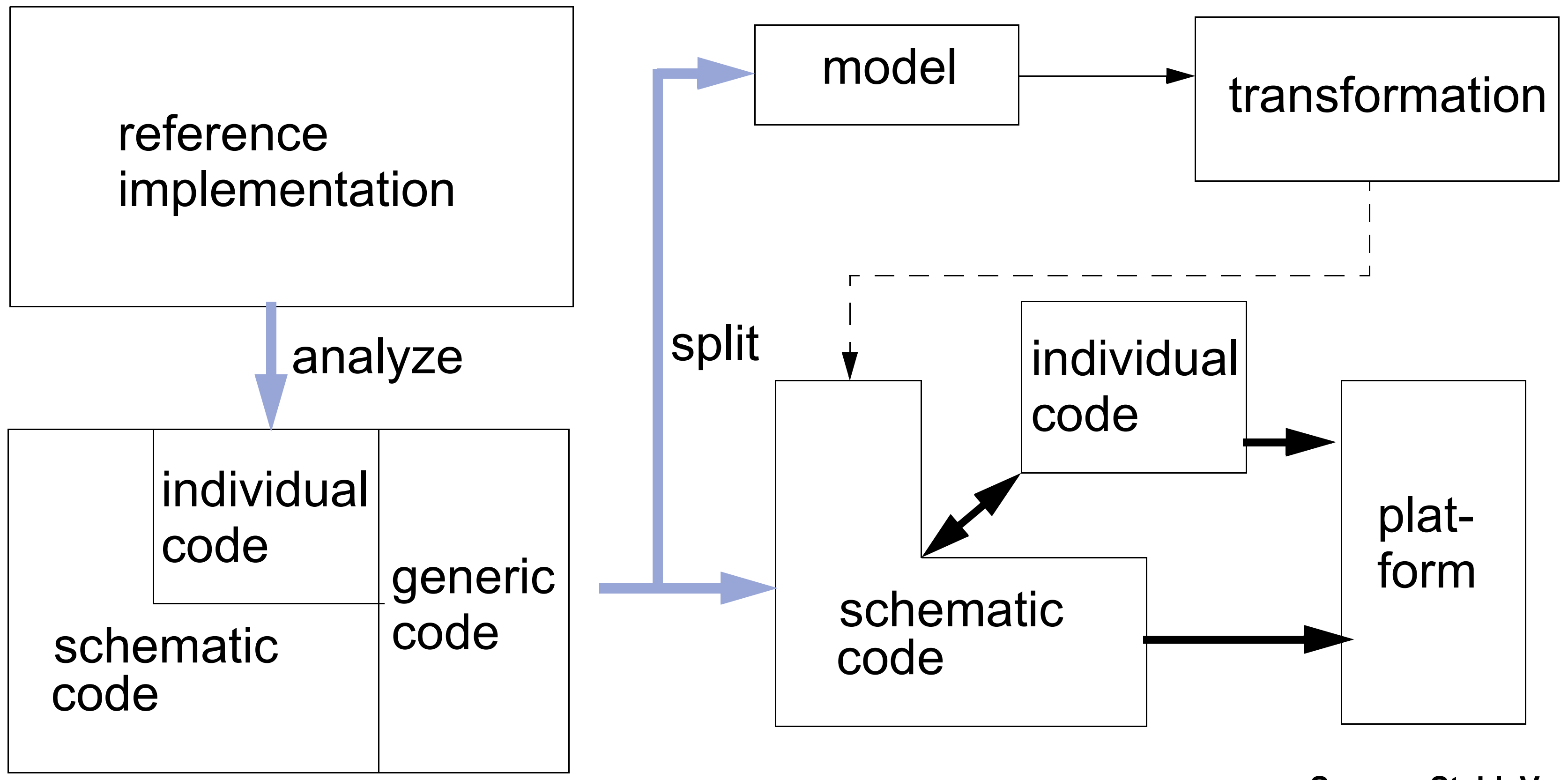
Model Driven Software Development (MDSD)

- Partial or whole generation of programs, based on a formal model
- Model represents the problem space of the application
- Models could be transformed in other models or into source code
- Model representation:
  - Abstract and formal description (without implementation details) of a problem space
  - Notation:
    - code
    - text
    - xml
    - graphical representation

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Concept of MDSD



Source: Stahl, Voelter, 2005

What can be Generated?

- Database Schema
- Data Access Layer
- User Interfaces
- Whole or part of the application logic
- Documentation
- Configurations
- Tests
- Wrapper
- Import/Export modules
- ...

Advantages of Software Generation

- Higher productivity
    - Tedious parts can be automated
    - Reduced reaction time on design changes/change requirements

- Improved quality
    - The transformation (template) is responsible for the quality of the code
    - Integrated architecture in templates defined
    - Automatic transformations (no careless errors)

Advantages of Software Generation

- Higher abstraction
  - Model represents an abstract description of the application
  - Business rules can be reviewed (or evan written) by domain experts
  - easier change to new technology (change templates)
  - better handling of complexity (reduction to essential)

- consistency of application
  - code generated based on rules is very consistent (naming conventions, parameter passing, ...) and so easy to understand and use
  - cross cutting concerns bundled in a central place (template/rule)

Disadvantages of Software Generation

- You have to learn a new technology and tools
- Increased (one-time) effort for
    - Developing/providing the infrastructure for model-driven software development
    - Development of transformation rules, templates for code generation
    - possibly development of a domain language (DSL) for the concrete application field field (--> software family)

Lightweight Software Generation

- We don’t use a general purpose Software generator Tool
- We do not have to spend a long time learning a complex tool and additional techniques like MOF, XMI, QVT, ATL, ...
- We write our own generator tool with minimal effort and maximum fit to our problem
- In this tutorial we use the following well established tools:
  - PHP
  - Regular Expressions
  - make
  - Bash coreutils like sed

- We typically achieve a lower level of generated code, but also spend much less effort to achieve this.

Motivating Example

Livedemo http://localhost/icwe ...

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Questions

- What distinguishes the two applications
  - Different Entitytypes (Person vs. Conference)
  - Different Attributes

- What do the two applications have in common
  - Functionality (CRUD GUI)
  - Architecture consisting of
    - CRUD Layer (OR-Mapper)
    - Controler
    - Views

- What could the models of the two applications look like?
  -> let’s take a look inside the code ...

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

CRUD-Layer (1)

- OR Mapper:

```
class CRUD_Person {
    protected $id;
    protected $firstname;
    protected $lastname;
    protected $birthday;

    function __construct($dictionary) {
        $this->id = set('id', $dictionary);
        $this->updateFields($dictionary);
    }

    function updateFields($dict) {
        $this->firstname = set('firstname', $dict);
        $this->lastname = set('lastname', $dict);
        $this->birthday = set('birthday', $dict);
    }

    function getId() {
        return $this->id;
    }
```

- Database Schema

```
drop table if exists Person;

create table Person (
   id integer auto_increment,
   firstname text,
   lastname text,
   birthday date,
   primary key(id)
);
```

CRUD-Layer (2)

```
function setFirstname($value) {
    $this->firstname = $value;
}

function getFirstname() {
    return $this->firstname;
}

function save() {
    $paras = [$this->firstname, $this->lastname, $this->birthday];
    if ($this->id) {
        $sql = "update Person
                   set firstname = ?, lastname = ?, birthday = ?
                 where id=?";
        $paras[] = $this->id;
    } else {
        $sql = "insert into Person
                values(null, ?, ?, ?)";
    }
    $ar = PDO_Util::query($sql, $paras);
   if (! $this->id)
      $this->id = PDO_Util::getLastID();
}
```

CRUD-Layer (3)

```
    static function getById($id) {
        $sql = "select * from Person where id = ?";
        $result = PDO_Util::query($sql, [$id]);
        if (count($result) != 1)
            die("No Person with id  '$id' found");
        return new Person($result[0]);
    }

    function delete() {
        $sql = "delete from Person where id = ?";
        $result = PDO_Util::query($sql, [$this->id]);
    }

} # End class
```

Hochschule Karlsruhe
University of
Applied Sciences

What is class specific ?

- OR-Mapper:

```
class CRUD_Person {
    protected $id;
    protected $firstname;
    protected $lastname;
    protected $birthday;

    function __construct($dictionary) {
        $this->id = set('id', $dictionary);
        $this->updateFields($dictionary);
    }

    function updateFields($dict) {
        $this->firstname = set('firstname', $dict);
        $this->lastname = set('lastname', $dict);
        $this->birthday = set('birthday', $dict);
    }

    function getId() {
        return $this->id;
    }
```

- Database Schema

```
drop table if exists Person;

create table Person (
   id integer auto_increment,
   firstname text,
   lastname text,
   birthday date,
   primary key(id)
);
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

What is class specific ?

```
function setFirstname($value) {
    $this->firstname = $value;
}

function getFirstname() {
    return $this->firstname;
}

function save() {
    $paras = [$this->firstname, $this->lastname, $this->birthday];
    if ($this->id) {
        $sql = "update Person
                   set firstname = ?, lastname = ?, birthday = ?
                  where id=?";
        $paras[] = $this->id;
    } else {
        $sql = "insert into Person
                values(null, ?, ?, ?)";
    }
    $ar = PDO_Util::query($sql, $paras);
   if (! $this->id)
      $this->id = PDO_Util::getLastID();
}
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

What is class specific ?

```
    static function getById($id) {
        $sql = "select * from Person where id = ?";
        $result = PDO_Util::query($sql, [$id]);
        if (count($result) != 1)
            die("No Person with id  '$id' found");
        return new Person($result[0]);
    }

    function delete() {
        $sql = "delete from Person where id = ?";
        $result = PDO_Util::query($sql, [$this->id]);
    }

} # End class
```

Observation

- Code of different applications/classes/views/controlers/... has many parts in common.
- Again: What could the model, describing the application, look like?

Observation

- Code of different applications/classes/views/controlers/... has many parts in common.
- Again: What could the model, describing the application, look like?
  - Application 1:

    ```
    <class: Person (firstname:text, lastname:text, birthday:date)>
    ```
  - Application 2:

    ```
    <class: Conference (name:text, abbreviation:text, country:text, city:text,\
                        location:text, year:integer)>
    ```

Observation

- Code of different applications/classes/views/controlers/... has many parts in com-mon.
- Again: What could the model, describing the application, look like?
    - Application 1:

      ```
      <class: Person (firstname:text, lastname:text, birthday:date)>
      ```

    - Application 2:

      ```
      <class: Conference (name:text, abbreviation:text, country:text, city:text,\
                          location:text, year:integer)>
      ```

- How could a datastructure for these models look like?

Model Datastructure

- Class Definition (model.php):

```
class MyModel {
    public string $name;
    public array $classes = [];

    # methods ...
}

class MyClass {
    public string $name;
    public array $attributes = [];

    # methods ...
}

class MyAttribute {
    public string $name;
    public string $type;

    # methods ...
}
```

- Example Instantiation (visualisation with `print_r()`)

```
MyModel Object
(
  [name] => Example1
  [classes] => Array (
     [0] => MyClass Object(
              [name] => Person
              [attributes] => Array(
                 [0] => MyAttribute Object (
                          [name] => firstname
                          [type] => text
                        )
                 [1] => MyAttribute Object (
                          [name] => lastname
                          [type] => text
                        )
                 [2] => MyAttribute Object (
                          [name] => birthday
                          [type] => date
                        )
              )
           )
)
```

Build model programatically

- Add some methods to build model:
- Model::addClass($name)
- Class::addAttribute($attribute_definition);

- Example:

```
$model = new MyModel('Example1');

$c = $model->addClass('Person');
$c->addAttribute('firstname:text');
$c->addAttribute('lastname:text');
$c->addAttribute('birthdate:date');

print_r($model);
```

name type

- Example Instantiation (visualisation with `print_r()`)

```
MyModel Object
(
  [name] => Example1
  [classes] => Array (
      [0] => MyClass Object(
               [name] => Person
               [attributes] => Array(
                  [0] => MyAttribute Object (
                           [name] => firstname
                           [type] => text
                         )
                  [1] => MyAttribute Object (
                           [name] => lastname
                           [type] => text
                         )
                  [2] => MyAttribute Object (
                           [name] => birthday
                           [type] => date
                         )
               )
           )
)
```

Template

```
# file: CRUD.tpl
# $model must conatain the model
#
<?php foreach ($model->classes as $class) { ?>

class CRUD_<?= $class->name ?> {
      protected $id;
  <?php foreach ($class->attributes as $att) { ?>
      protected $<?= $att->name ?>;
  <?php } ?>

  function __construct($dictionary) {
      $this->id = set('id', $dictionary);
      $this->updateFields($dictionary);
  }

  function updateFields($dictionary) {
      <?php foreach ($class->attributes as $a) { ?>
          $this-><?= $a->name ?> = set('<?= $a->name ?>', $dictionary);
      <?php } ?>
  }

  function getId() {
      return $this->id;
  }
```

Iterate over each class in model

Iterate over all attributes of a class

Template

```
<?php foreach ($class->attributes as $a) { ?>
     function set<?= ucFirst($a->name) ?>($value) {
        $this-><?= $a->name ?> = $value;
      }

     function get<?= ucFirst($a->name) ?>() {
        return $this-><?= $a->name ?>;
      }
 <?php } ?>

 static function getById($id) {
      $sql = "select * from <?= $class->name ?> where id = ?";
      $result = PDO_Util::query($sql, [$id]);
      if (count($result)!=1)
          die("No <?= $class->name ?> with id  '$id' found");
      return new <?= $class->name ?>($result[0]);
   }

  # ...


}
<?php } ?>
```

Iterate over all attributes of a class to generate getter and setter methods

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Generator

- File: generator.php

```
<?php

include 'model.php';

if (count($argv) < 3)
    die("usage: $argv[0] <model-file> <template-file>\n");

$model= MyModel::readModelFile($argv[1]);
include $argv[2];
```

code formatter

generated OR-Layer

- usage:

```
$ php generator.php Example1.model CRUD.tpl | php_prettyprinter > CRUD.php
```

```
<class: Person (firstname:text, lastname:text, birthday:date)>
```

Generator

- File: generator.php

```
<?php

include 'model.php';

if (count($argv) < 3)
    die("usage: $argv[0] <model-file> <template-file>\n");

$model= MyModel::readModelFile($argv[1]);
include $argv[2];
```

generated OR-Layer

code formatter

- usage:

```
$ php generator.php Example1.model CRUD.tpl | php_prettyprinter > CRUD.php
```

```
<class: Person (firstname:text, lastname:text, birthday:date)>
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Model Transformation

```
class MyModel {
    public string $name;
    public array $classes = [];

    # constructor & outher methods ...

    static function readModelFile($file) {
        $model = new MyModel($file);
        $lines = file($file, FILE_IGNORE_NEW_LINES);
        foreach ($lines as $line) {
            if (preg_match('#<class:\s*(\w+)\s*\((.*)\)\s*>#',$line, $match)) { # parse line from model
                $class_name = $match[1];
                $attributes = $match[2];
                $att_list = preg_split("#\s*,\s*#", $attributes);
                $c = $model->addClass($class_name);
                foreach ($att_list as $a) {
                    $c->addAttribute($a);
                }
            } else {
                if (trim($line))
                  throw new Exception("illegal line '$line' in file '$file' detected");
            }
        }
        return $model;
    }
```

regular expressions

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Regular Expressions

- Powerful text pattern language
- Allows the filtering/substitution of text patterns
- Implementation in many computer languages
- consists of
  - literal characters (A...Z, a...z 0...9 _, ...)
  - meta characters ([ ] ( ) { } | ? + - * ^ $ \ . \b)
  - character classes:
    - predefined: \w \d \s \W \D \S
    - user defined: [A-Z] [aeiou] [0-9A-Fa-f] ...

- Predifined character classes:

```
\w: word character
\d: digit
\s: whitespace, <tab>
\W: inverse of \w
\D: inverse of \D
\S: inverse of \s
```

Hochschule Karlsruhe
University of
Applied Sciences

Concepts

- Quantifier: define how many times the token before (or bracketed expression) should be matched
  - * : zero or more (greedy)
  - *? : zero or more (ungreedy or lazy)
  - + : one or more (greedy)
  - +? : one or more (ungreedy or lazy)
  - ? : zero or one
  - {3,5} : three to five
  - {3,} : three or more
  - {,5} : less or equal five

Concepts

- Backreferences: if parts of a matched text should be used later, use round brak-kets to mark these parts (referenced[1] later by \1, \2, \3, ...)
- Also use round brackets to group expressions (i.e. together with quantifiers)
- Position in pattern
    - ^ : Start of pattern
    - $: End of pattern
    - \b : Word boundary

1. or $1, $2, $3, ... depening on used tool/implementation

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Examples of Regular Expressions

- Matching:
  - IP-Address: `([0-9]{1,3}\.){3}[0-9]{1,3}`
  - Number between 100 and 9999 : `[1-9][0-9]{2,3}`
  - Extract headings from a HTML-Document: `<h([0-3])>(.*?)</h\1>`
  - A number (i.e. 1, -2.564, 0.1, ...) : `-?\d+(\.\d+)?`

- Replacing (perl syntax)
  - Change your winter hobby: `s#\bSki\b#Snowboard#g`
  - Remove markup from HTML: `s#<.*?>##g`
  - Make Hyperlink from URL:
    `s#\b(https?://(.*?))\s#<a href="\1">\2</a>#g`

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Embedding Regexes in a Programming Language

- The perl & sed way
  - Regexes are integral part of the language
  - Examples (Perl):

```
$text = "The computers with ip-addresses 123.34.45.234 and 123.34.32.1 are infected";

if ($text =~ /((\d{1,3}\.){3}\d{1,3})/) {
  print "IP-address $1 found\n";
}
print "All IP-Addresses:\n";
while ($text =~ /((\d{1,3}\.){3}\d{1,3})/g) {
  print $1,"\n";
}

$text = "The URL of my institute is https://www.iai.kit.edu";
$text=~s#\b(https?://([\w./]+))#<a href="$1">$2</a>#g;
print $text;
```

- Output:

```
IP-Address 123.34.45.234 found

All IP-Addresses:
123.34.45.234
123.34.32.1

The URL of my institute is
<a href="https://www.kit.edu">\
www.kit.edu</a>"
```

Embedding regexes in a Programming Language

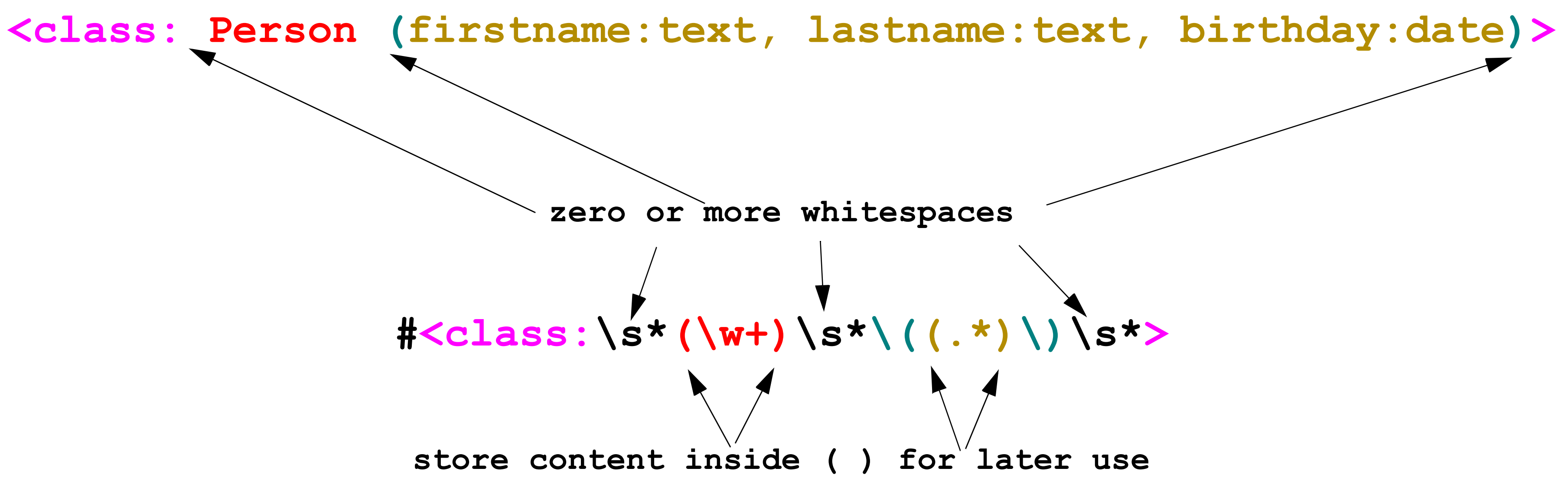
- The other way (e.g. PHP, Java, Python, ...)
  - Integration via library
  - Regular expressions are handled as Strings
  - Examples (PHP)

```
$text = "The computers with ip-addresses 123.34.45.234 and 123.34.32.1 are infected";

if (preg_match('/((\d{1,3}\.){3}\d{1,3})/',$text, $match)) {
  print "IP-address ".$match[1]." found\n";
}

print "All ip-addresses:\n";
if (preg_match_all('/((\d{1,3}\.){3}\d{1,3})/',$text, $all_matches))
  foreach ($all_matches[0] as $match)
    print "$match\n";

$text = "The URL of my institute is http://www.iai.fzk.de";
$text= preg_replace ('#\b(http://([\w./]+))#','<a href="\1">\2</a>', $text);
print $text;
```

Regular Expression for Model Description

- Code:

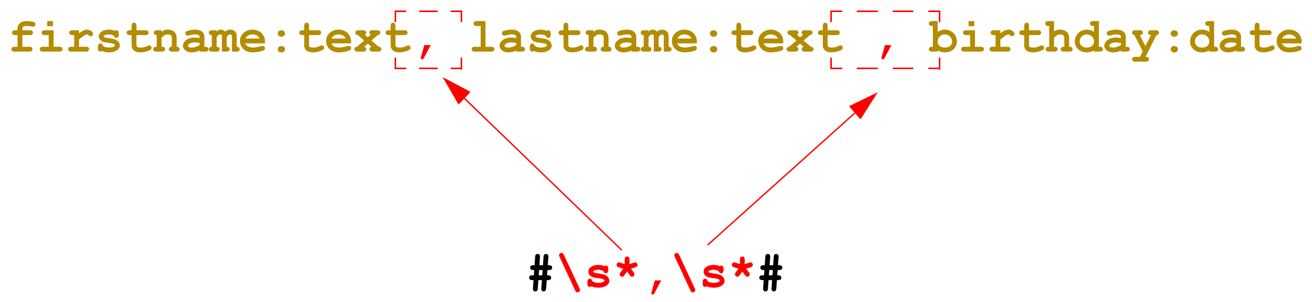
```
if (preg_match('#<class:\s*(\w+)\s*\((.*)\)\s*>#',$line, $match)) {
     $class_name = $match[1];
     $attributes = $match[2];
     ...
}
```


<class: Person (firstname:text, lastname:text, birthday:date)>
zero or more whitespaces
#<class:\s*(\w+)\s*\((.*)\)\s*>
store content inside ( ) for later use

Regular Expression for Model Description

- Code:

```
if (preg_match('#<class:\s*(\w+)\s*\((.*)\)\s*>#',$line, $match)) {
     $class_name = $match[1];
     $attributes = $match[2];
     $att_list = preg_split("#\s*,\s*#", $attributes);
     ...
}
```



```
firstname:text, lastname:text , birthday:date

#\s*,\s*#
```

- result:

```
$att_list = ['firstname:text','lastname:text', 'birthday:date']
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Exercise I

- Minimal generator framework consisting of:
  - generator `exercise1.php`
  - Metamodel: `model.php`
  - Model `exercise_1.model`
  - Template `show-model.tpl`
- Tasks:
  - Install XAMPP (if not already done)
  - Customize the template file, so that valid MySQL DDL-code is generated
  - Extend meta-model file, so that further datatypes are accepted.

URL: https://www.smiffy.de/icwe-2022/exercise-I.pdf

Part II - Software Generator Models

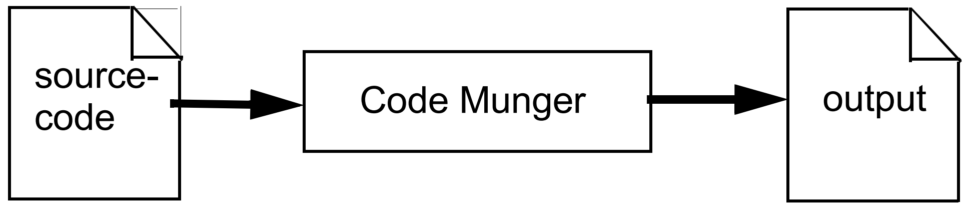
- Code Munger
- Inline Code Expander
- Mixed Code Generator
- Partial Class Generator
- Full Tier Generator
- Domain Specific Language

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Code Munger

- Workflow



- Functionality:
  - Input is source code
  - Extraction of relevant aspects from the code
  - Transformation to output format
  - Extraction is commonly based on regular expressions
- Examples:
  - Javadoc
  - Transformation from one DDL-dialect to another (i.e. MySQL -> Oracle)
  - Code stub generation

Example: Code Munger

- Oracle -> MySQL Dialect converter (for a concrete Schema):

```
CREATE TABLE City
(Name VARCHAR2(35),
 Country VARCHAR2(4),
 Province VARCHAR2(32),
 Population NUMBER CONSTRAINT CityPop
   CHECK (Population >= 0),
 Longitude NUMBER(5,2) CONSTRAINT CityLon
   CHECK ((Longitude >= -180) AND (Longitude <= 180)) ,
 Latitude NUMBER(5,2) CONSTRAINT CityLat
   CHECK ((Latitude >= -90) AND (Latitude <= 90)) ,
 CONSTRAINT CityKey PRIMARY KEY (Name, Country, Province));
```

```
CREATE TABLE City
(Name varchar(35),
 Country varchar(4),
 Province varchar(32),
 Population integer ,constraint CityPop
   CHECK (Population >= 0),
 Longitude float ,constraint CityLon
   CHECK ((Longitude >= -180) AND (Longitude <= 180)) ,
 Latitude float ,constraint CityLat
   CHECK ((Latitude >= -90) AND (Latitude <= 90)) ,
 PRIMARY KEY (Name, Country, Province));
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

- file: o2m.sed

delete oracle specific statements

transform datatypes

delete „CONSTRAINT <label>" directive

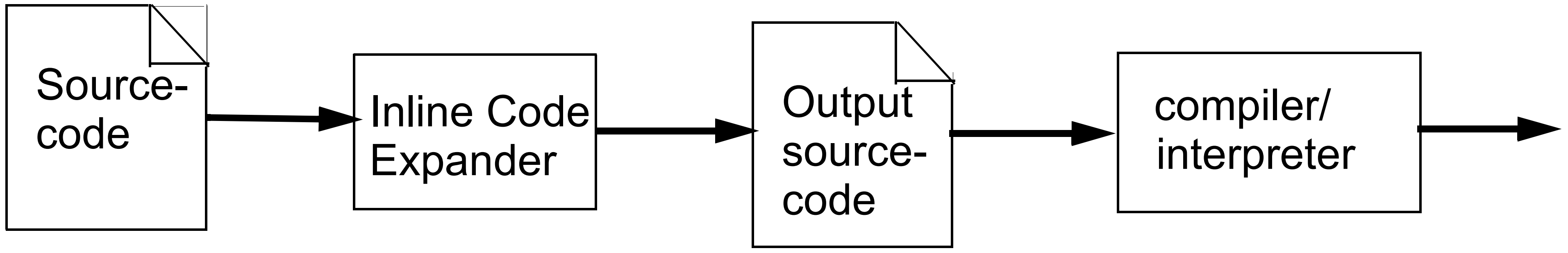
```
s#^ *whenever.*##i;
s#^ *alter session.*##i;
s#^ *quit;##;

s#varchar2#varchar#i;
s#number\([0-9]+, *[0-9]+\)#float#i;
s#number#integer#i;
s#CONSTRAINT +[A-Za-z0-9_]+ *(primary key|not null|unique)#\1#i;
s#CONSTRAINT +([A-Za-z0-9_]+)#,constraint \1#i;
```

add , before „constraint" statement

```
$ sed -E -f o2m.sed oracle-mondial-schema.sql > mysql-mondial-schema.sql
```

Inline Code Expander

- Workflow

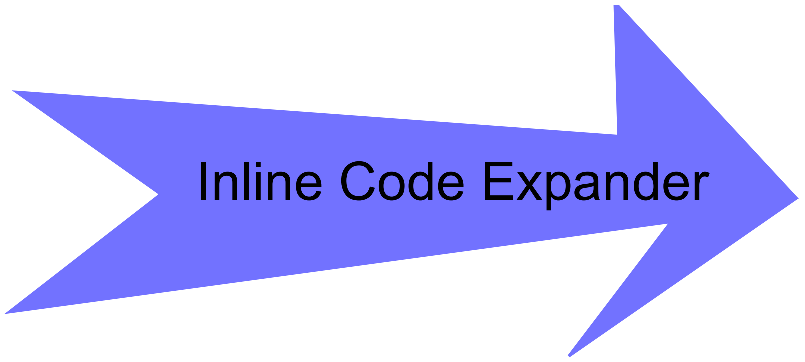


- Functionality:
  - Implicit definition of a new language (extension of existing language)
  - Simplifies the writing of source code
  - Output is input for a compiler/interpreter
- Examples:
  - Generation of classes from abstract description
  - Add logging, etc.

Example

- Sourcecode test.php.ice

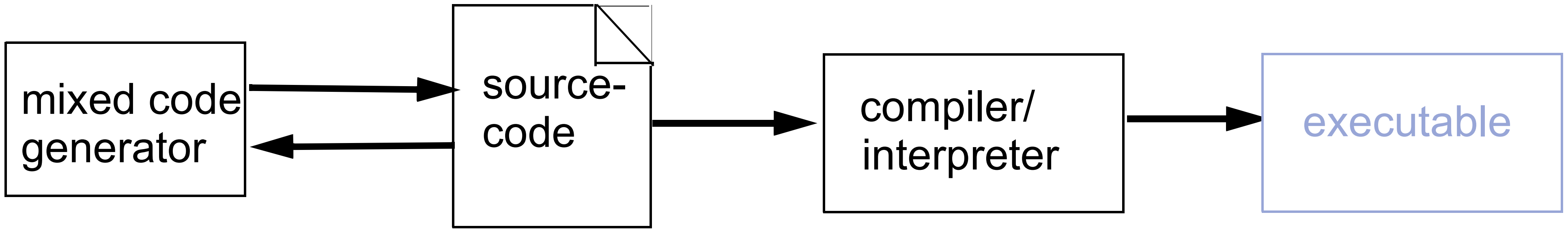
```
<?php

<class: Person (surname, firstname, birthday) >

<class: Film (title, year, regisseur) >

$p1 = new Person('Waits', 'Tom', '9.9.1949');
$p2 = new Person('Jarmusch', 'Jim', '22.1.1953');
$f1 = new Film('Short Cuts', 1989, $p2);

echo "A simple test:\n\n";
echo "{$p1->getSurname()} {$p1->getFirstname()} {$p1->getBirthday()}\n";
echo "{$f1->getTitle()} ({$f1->getYear()), director: {$f1->getDirector()->getLastname()\n";

?>
```



```php
<?php

    class Person { ... }

    class Film {
         protected $title;
         protected $year;
         protected $director;

        function  __construct($title,$year,$director) {
         $this->title = $title;
         $this->year = $year;
         $this->director = $director;
}

     function getTitle() {
         return $this->title;
     }

     function getYear() {
         return $this->year;
     }

     function getDirector() {
         return $this->director;
     }
   }

$p1 = new Person('Waits', 'Tom', '9.9.1949');
$p1 = new Person('Jarmusch', 'Jim', '22.1.1953');
$f1 = new Film('Short Cuts', 1989, $p2);
...
```

Mixed Code Generator

- Workflow



- Functionality
  - Special implementation of the inline code expander
  - The content of the input file is replaced by the output
  - special syntax is hidden as comments
- Examples
  - Like Inline code expander
  - ...

Mixed Code Generator - Example

- file: source.php

```
class person {
    protected $id;
    private $lastname;
    private $firstname;

   function __construct($id, $lastname,
                        $firstname){
        $this->id = $id;
        $this->lastname = $firstname;
        $this->firstname = $firstname;
   }

    // get($id)

    // get($lastname)
    // set($lastname)

    // set($firstname)
    // get($firstname)

}
```

```
class person {

    ...

    function getId() {
        return $this->id;
    }

    function getLastname() {
        return $this->lastname;
    }
    function setLastname($lastname) {
        $this->lastname = $lastname;
    }

    function setFirstname($firstname) {
        $this->firstname = $firstname;
    }
    function getFirstname() {
        return $this->firstname;
    }
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Mixed Code Generator - Implementation

- Implementation with regular expressions in perl (as command line tool)
  - perl command line options:
    - -p: assume while (<>) { ... } loop around code, print lines
    - -i.bak: in-place substitution
  - call:

```
perl -p -i.bak transformation.pl source.php
```

- file: transformation.pl

```
s#^(\s*)//\s*set\(\$(\w+)\)#
$1function set\u$2(\$$2) {
$1    \$this->$2 = \$$2;
$1}#;

s#^(\s*)//\s*get\(\$(\w+)\)#
$1function get\u$2() {
$1    return \$this->$2;
$1}#;


# explanations:
# \u: uppercase next char (perl extension)
# $1: indent (whitespaces)
# $2: name of variable
# \$: print a '$'-character
# \(: matches a '('-character
```

Partial Class Generator

- Functionality:
  - based on an explicit definition file (an abstract model)
  - generates a number of base classes
  - Manual extensions in derived classes or „protected areas“
  - Initial point for building a „Tier Generator“

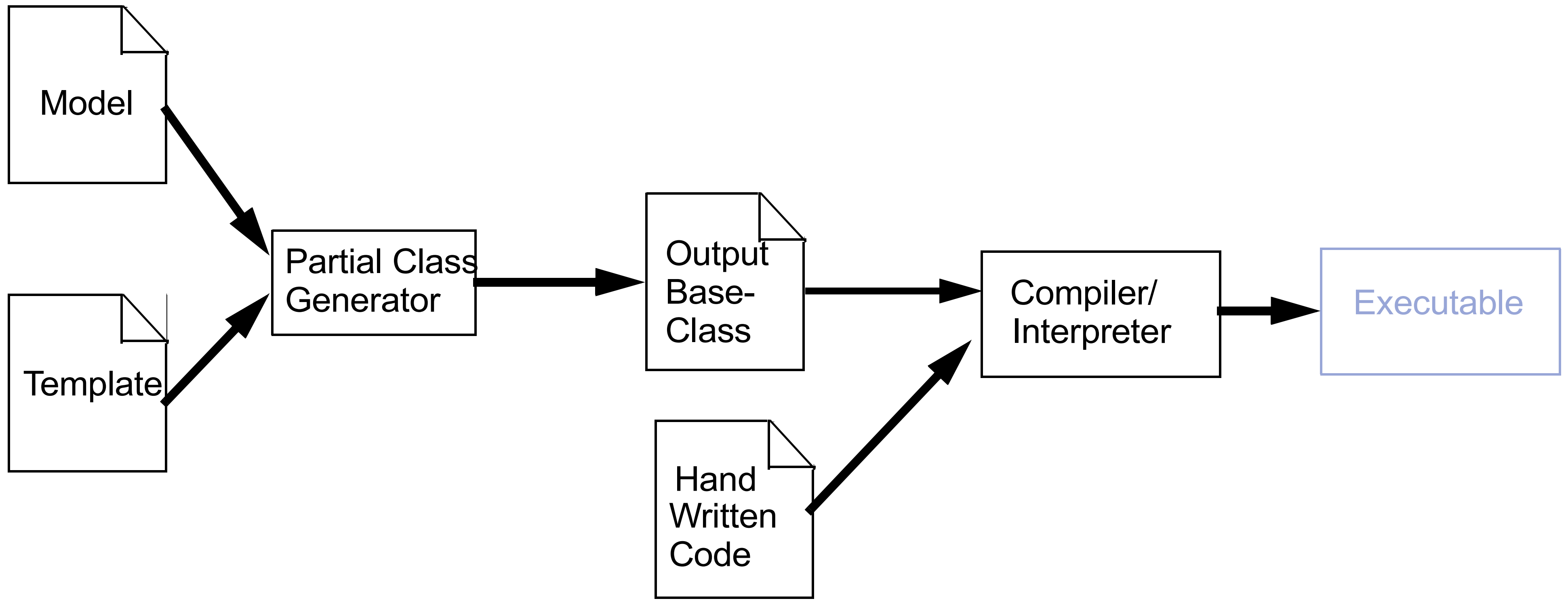
- Examples:
  - Data access layer
  - Database schema
  - User interfaces
  - Import-/export filters

Karlsruhe Institute of Technology

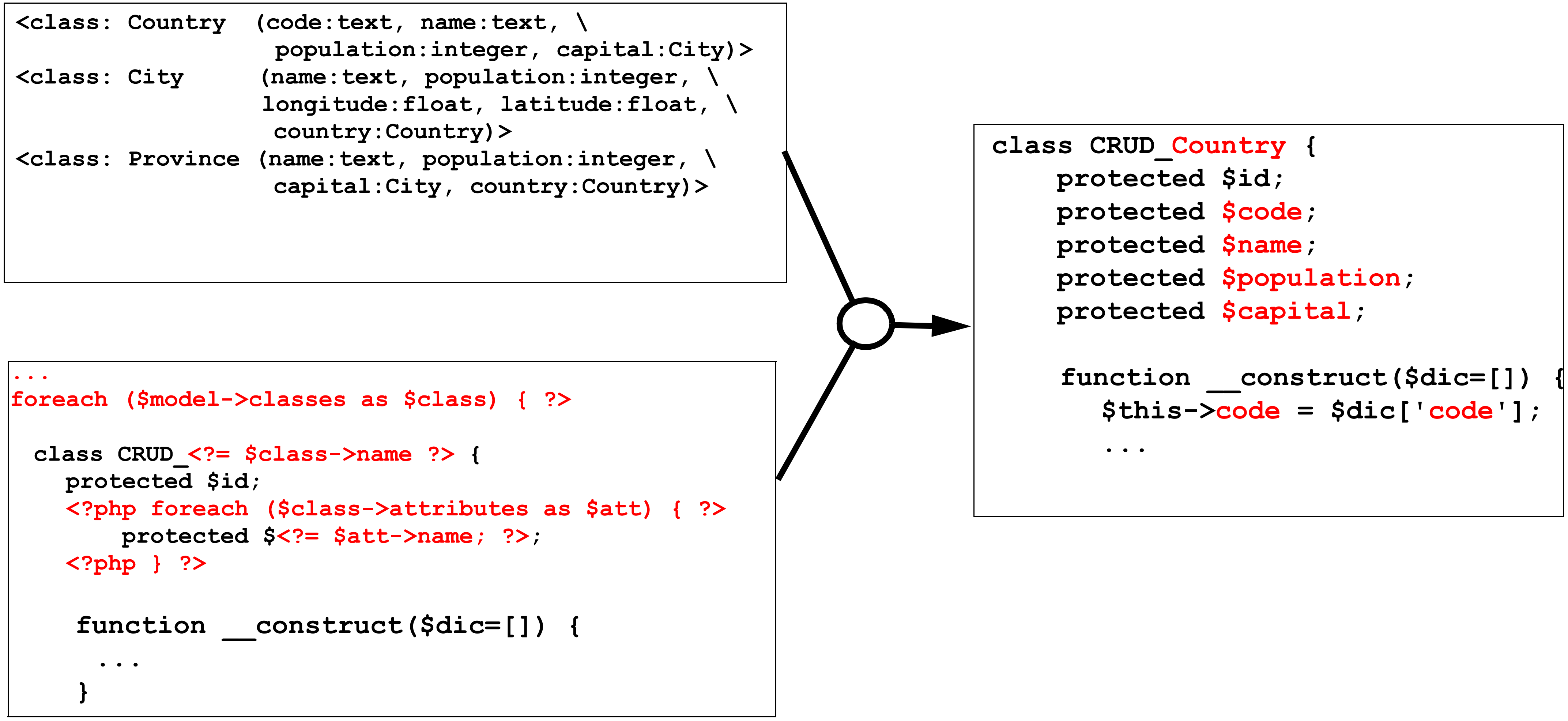
Hochschule Karlsruhe
University of
Applied Sciences

Partial Class Generator

- Workflow



Partial Class Generator



```
<class: Country  (code:text, name:text, \
                  population:integer, capital:City)>
<class: City      (name:text, population:integer, \
                  longitude:float, latitude:float, \
                  country:Country)>
<class: Province (name:text, population:integer, \
                  capital:City, country:Country)>
```

```
...
foreach ($model->classes as $class) { ?>

 class CRUD_<?= $class->name ?> {
   protected $id;
   <?php foreach ($class->attributes as $att) { ?>
       protected $<?= $att->name; ?>;
   <?php } ?>

    function __construct($dic=[]) {
      ...
    }
```

```
class CRUD_Country {
    protected $id;
    protected $code;
    protected $name;
    protected $population;
    protected $capital;

    function __construct($dic=[]) {
       $this->code = $dic['code'];
       ...
```

Karlsruhe Institute of Technology

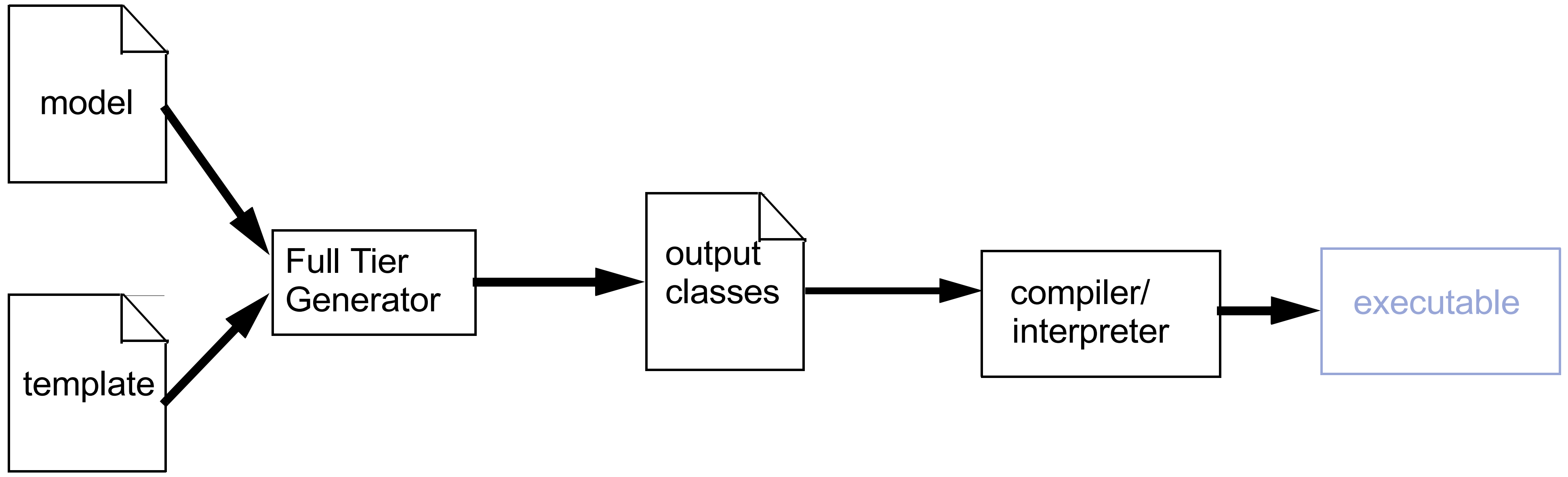
Hochschule Karlsruhe
University of
Applied Sciences

Tier Generator Model

- Functionality
  - Like „Partial Class Generator“, but it generates the code of a tier of an application
  - Whole application logic outside of codebase
  - „Partial Class Generator“ is a good starting point for building a „Tier Generator model“
- Examples
  - Database Access layer
  - Web client layer
  - Data Import/Export

Tier Generator Model - Workflow

- Workflow

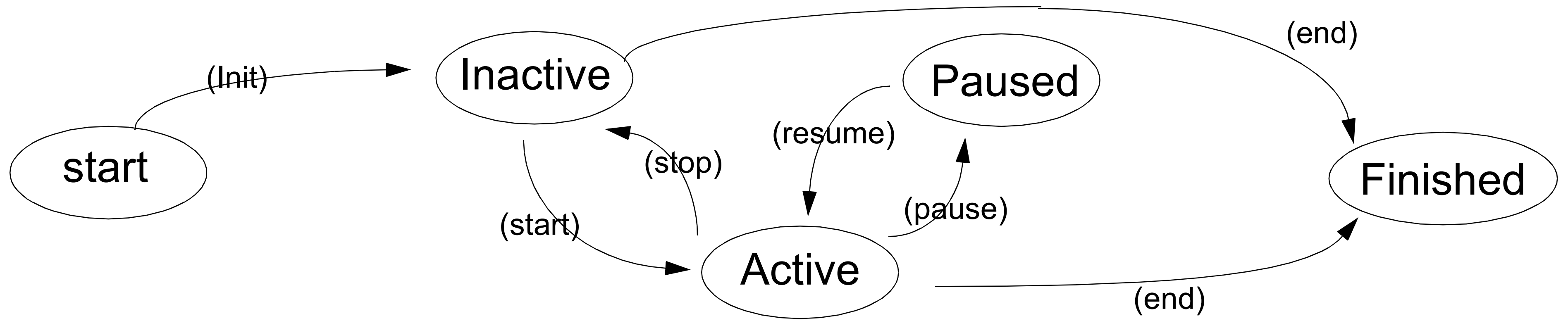


Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Tier Generator Model - Example

- State Diagram



- Model File:

```
start   -(init)-> Inactive
                  Inactive -(start)-> Active
                  Inactive <-(stop)-  Active
                  Inactive  -(end)->                              Finished
                                      Active  -(pause)->   Paused
                                      Active  <-(resume)-  Paused
                                    Active  -(end)->                Finished
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Tier Generator Model - Internal Model

```
class Statechart {

  public array $states = [];



  // inserts a new transition and, if not already
   // known, the start and end state of this
   // transition are also created
   function createTransition($s0,
                             $event,
                             $s1) {

       ...
   }

  // returns an array with all states (type State)
   function getStates() {
      ...
   }

   // returns the name of the start state
  function getStartState() {
      ...
}
```

```
class State {

  public $name;

  # transitions dictionary:
  #   key: event,  value: State-Instance
  public $transitions = [];

  function __construct($name) {
    $this->name = $name;
  }

   // adds transition
   //      $this -($e)-> $state
   //
   function addTransition($e, $state) {
       ...
   }

   // retuns a possible transition event
   function getRandomEvent() {
      ...
   }

}
```

Tier Generator Model - Template

```
  ...
function transition($event) {
    <?php foreach ($model->getStates() as $state) { ?>
        if ($this->actual_state == '<?= $state->name ?>') {
            <?php foreach ($state->getTransitions() as $t_event=>$t_state) { ?>
                if ($event=='<?=$t_event ?>')
                    $new_state = '<?php echo $t_state->name ?>';
              else
             <?php } ?>
             die("Illegal event ($event) in state '$this->actual_state'");
        } else
    <?php } ?>
    die("statemachine is in unknowm state ($this->actual_state)");
    $this->actual_state = $new_state;

    return $new_state;
}
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Tier Generator Model - Generated Code

```
function transition($event) {
       if ($this->actual_state == 'start') {
           if ($event == 'init') $new_state = 'Inactive';
           else die("Illegal event ($event) in state '$this->actual_state'");
       } else if ($this->actual_state == 'Inactive') {
           if ($event == 'start') $new_state = 'Active';
           else if ($event == 'end') $new_state = 'Finished';
           else die("Illegal event ($event) in state '$this->actual_state'");
       } else if ($this->actual_state == 'Active') {
           if ($event == 'stop') $new_state = 'Inactive';
           else if ($event == 'pause') $new_state = 'Paused';
           else if ($event == 'end') $new_state = 'Finished';
           else die("Illegal event ($event) in state '$this->actual_state'");
       } else if ($this->actual_state == 'Paused') {
           if ($event == 'resume') $new_state = 'Active';
           else die("Illegal event ($event) in state '$this->actual_state'");
       } else if ($this->actual_state == 'Finished') {
           die("Illegal event ($event) in state '$this->actual_state'");
       } else die("statemachine is in unknowm state ($this->actual_state)");
       $this->actual_state = $new_state;
       return $new_state;
   }
```

Domain Specific Language (DSL)

- Functionality
  - A language, which is closely related to your problem domain
  - Used to build the application logic of a program in that domain
  - Special tools to build the parser for your language (lex, yacc, bison, antlr, ...)

- Examples
  - mathematica, matlab
  - make, ant
  - SQL
  - ...

Exercise II

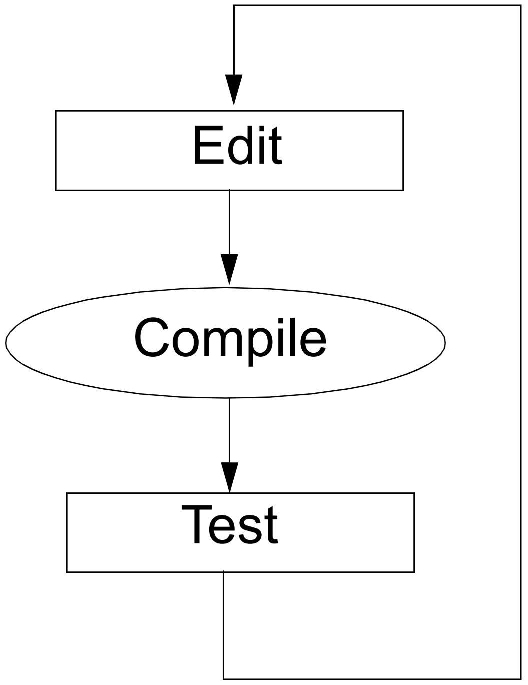
- Minimal generator framework consisting of:
  - generator `exercise-3.php`
  - Metamodel: `statechart-model.php`
  - Model `exercise_3.model`
  - Template `show-model.tpl`
- Tasks:
  - Implementation of the static method `Statechart::importModel($file)` that transform the external (ascii based) model description into the internal model representation of the generator.
  - Homework:
    - Adapt the template file, so that the code from slide 59 is generated.
    - Use the method `Statechart::getRandomEvent($actual_state)` to test your generated code

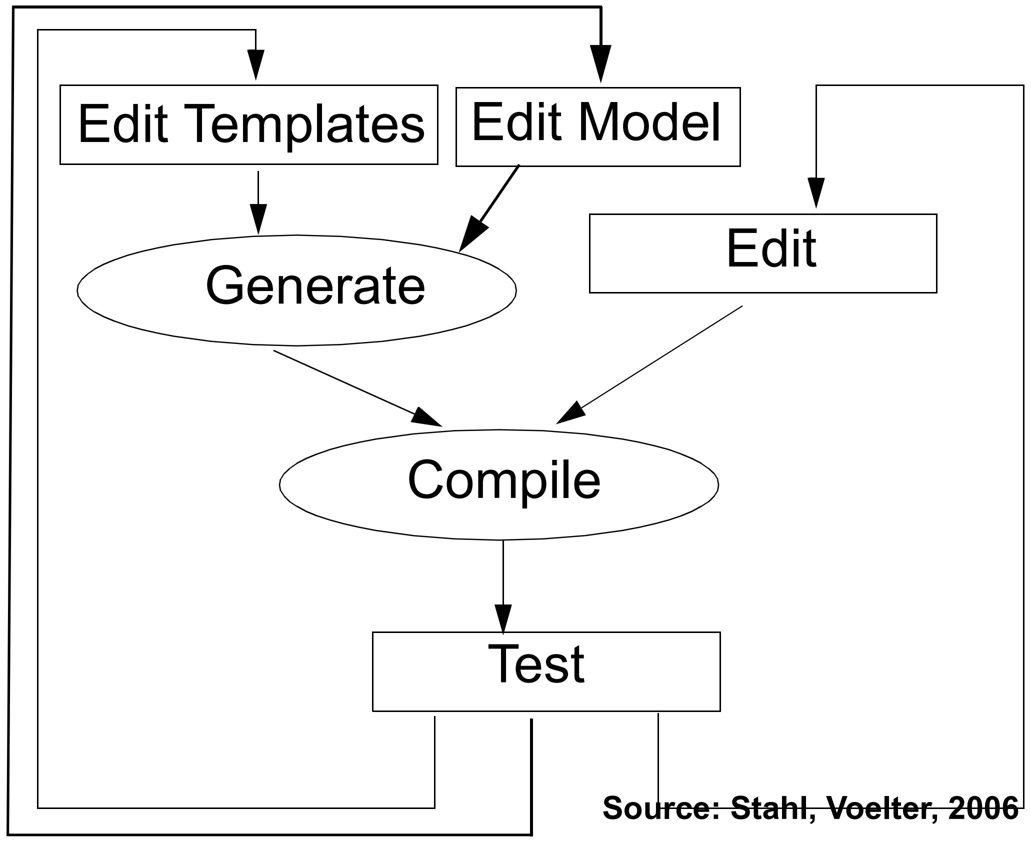
Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Software Development: Comparision of Workflow

- Traditional Software Engineering
- MDSD





Source: Stahl, Voelter, 2006

Workflow

- Back to or motivation Example (Person, Conference model)
- Files:
  - Generator
    - generator.php
    - model.php
  - Model files:
    - Example1.model
    - Example2.model
  - Templates:
    - ApplicationClass.tpl
    - controler.tpl
    - CRUD.tpl
    - edit.tpl
    - list.tpl
    - schema.mysql.tpl

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Build process

```
MODEL=Example1.model
# Database schema and OR-layer:
php generator.php $MODEL schema.mysql.tpl > schema.mysql.ddl
php generator.php $MODEL CRUD.tpl | php_prettyprint > OR_Base.php

# application logic (only if not existent)
sed -E 's/<class: *([A-Za-z]+) *.*/\1.php/' $(MODEL) | grep -v '^ *$$' > .class
if [ ! -f `cat .class` ] ;
then
    php generator.php $MODEL ApplicationClass.tpl > `cat .class` ;
fi

# Web interface (views, Controler)
php generator.php $MODEL list.tpl > list.php

php generator.php $MODEL controler.tpl > controler.php
php generator.php $MODEL edit.tpl  > edit.php
```

Dependencies

```
MODEL=Example1.model
# Database schema and OR-layer:
php generator.php $MODEL schema.mysql.tpl > schema.mysql.ddl
php generator.php $MODEL CRUD.tpl | php_prettyprint > OR_Base.php

# application logic (only if not existent)
sed -E 's/<class: *([A-Za-z]+) *.*/\1.php/' $(MODEL) | grep -v '^ *$$' > .class_file
if [ ! -f `cat .class_file` ] ;
then
    php generator.php $MODEL ApplicationClass.tpl > `cat .class_file` ;
fi

# Web interface (views, Controler)
php generator.php $MODEL list.tpl > list.php

php generator.php $MODEL controler.tpl > controler.php
php generator.php $MODEL edit.tpl  > edit.php
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Unix tool make

- „In software development, Make is a build automation tool that automatically builds executable programs and libraries from source code by reading files called Makefiles which specify how to derive the target program“ [wikipedia].
- A makefile consists of a number of rules:

```
target:    dependencies ...
           commands
           ...
```

  - Example:

```
schema.mysql.ddl:  schema.mysql.tpl Example1.model
             php generator.php Example1.model \
                               schema.mysql.tpl >  schema.mysql.ddl
```

  - Timestamps of files decide if a target is already fullfilled

Automation with make

```
app=1
MODEL=Example$(app).model

build: OR_Base.php list.php controler.php edit.php app-class .db
        echo rebuild successful

OR_Base.php: gnerator.php $(MODEL) CRUD.tpl
        php generator.php $(MODEL) CRUD.tpl | php_prettyprint > OR_Base.php
        php OR_Base.php

.db: schema.mysql.ddl
        mysql -u root -h 127.0.0.1 icwe < schema.mysql.ddl
        if [ -f data/$(MODEL).data ]; \
        then \
                mysql -u root -h 127.0.0.1 icwe < data/$(MODEL).data ; \
        fi
        touch .db

schema.mysql.ddl:  schema.mysql.tpl $(MODEL)
        php generator.php $(MODEL) schema.mysql.tpl | ddl_prettyprint > schema.mysql.ddl
```

targets on next page

Karlsruhe Institute of Technology

Hochschule Karlsruhe
University of
Applied Sciences

Automation with make

```
rebuild: clean build

clean:
        touch *.tpl

list.php: list.tpl $(MODEL)
        php generator.php $(MODEL) list.tpl > list.php

controler.php: controler.tpl $(MODEL)
        php generator.php $(MODEL) controler.tpl > controler.php

edit.php: edit.tpl $(MODEL)
        php generator.php $(MODEL) edit.tpl  > edit.php

app-class:
        sed -E 's/<class: *([A-Za-z]+) *.*/\1.php/' $(MODEL) | grep -v '^ *$$' > .class
        @if [ -f `cat .class` ] ; \
        then \
            php generator.php $(MODEL) ApplicationClass.tpl > `cat .class` ; \
        fi
```

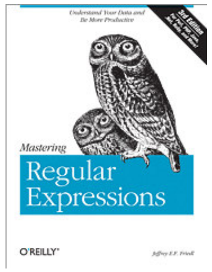
Conclusion

- Codegeneration deals with the partial or complete generation of programs, based on a formal model
- A model could be written in a specific language (model language), existing source code or also available meta information (Database Metadata, XML-Schema, ...)
- Regular Expressions are a powerful language to extract information from code or a formal model
- Lightweight software generators consist often only about a dozens of lines
- Code generation yields to higher abstraction, higher productivity, improved quality and a higher consistence of your application

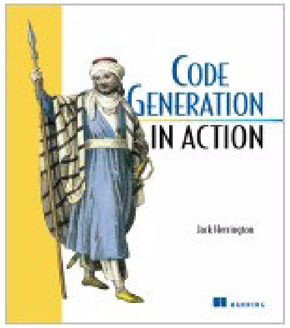
Resume: It’s much more interesting to write programs that write programs than to write programs oneself

Resources

- Jeffrey E. F. Friedl, Mastering Regular Expressions, Third Edition, O’Reilly, August 2006



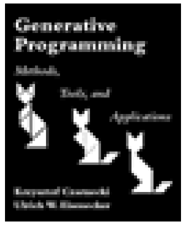
- Jack Herrington: *Code Generation in Action*. Manning Verlag, 2003, 350 Seiten, ISBN: 1930110979



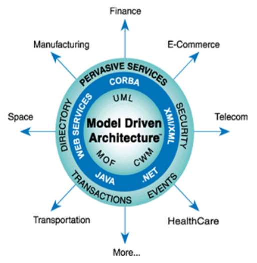
- http://www.codegeneration.net/

Resources

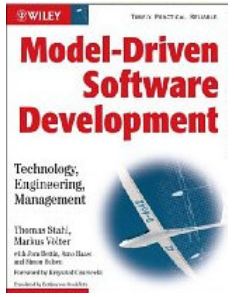
- Krzysztof Czarnecki, Ulrich Eisenecker: *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley Professional; 1. Auflage, 2000



- http://www.omg.org/mda/



- Markus Völter, Thomas Stahl: *Model-Driven Software Development - Technology, Engineering, Management.* Wiley & Sons, May 2006



- Homepage Markus Völter: http://www.voelter.de/services/mdsd.html