Rascal: a language for all things source code

Tijs van der Storm

CWI Centrum Wiskunde & Informatica

University of Amsterdam

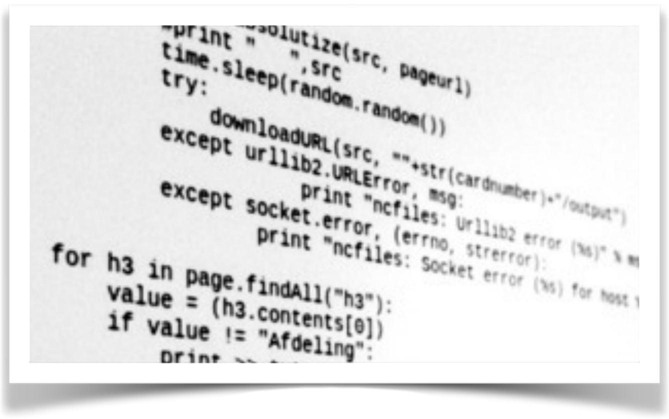
Meta talks

- Language design smells (London)
- Orthogonality in language design (Austin)
- Abstract syntax sucks! (Aarhus)
- I want my live programming (Skamania)
- Celldown: source code as UI (Athens)

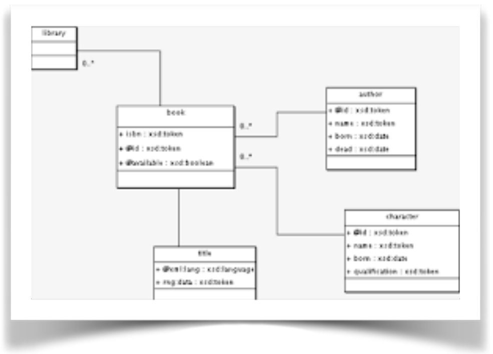
Today: Rascal

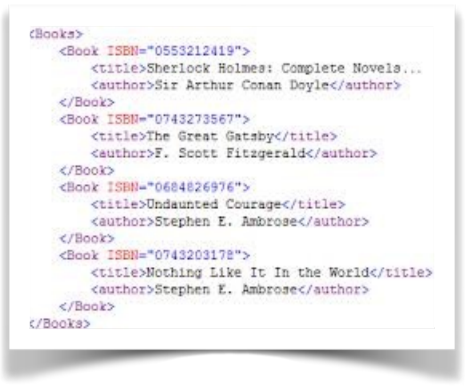
- So a concrete language,
- but for meta programming :-D

http://www.rascal-mpl.org



```
print "      ",src
time.sleep(random.random())
try:
    downloadURL(src, ""+str(cardnumber)+"/output")
except urllib2.URLError, msg:
        print "ncfiles: Urllib2 error (%s)" % m
except socket.error, (errno, strerror):
        print "ncfiles: Socket error (%s) for host
for h3 in page.findAll("h3"):
    value = (h3.contents[0])
    if value != "Afdeling":
```





```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Facts about source code

Generated source code

Refactored source code

Visualized source code

In brief…

- “Functional” programming language
- Data types: <the usual>, sets, relation, parse tree, source location
- Built-in context free grammar formalism
- Pattern matching: list (A), set (ACI), deep, concrete syntax
- Traversal primitive “visit”

Applications

- Source code analysis
  - PHP [ASE’15, ASE’14, ISSTA’13]
  - Java/C [ICSME’14, JSEP’16]
- DSLs
  - Derric: digital forensics [ICSE’11]
  - Machinations: game economies [SLE’13, FDG’14]
  - Rebel: banking (ING)

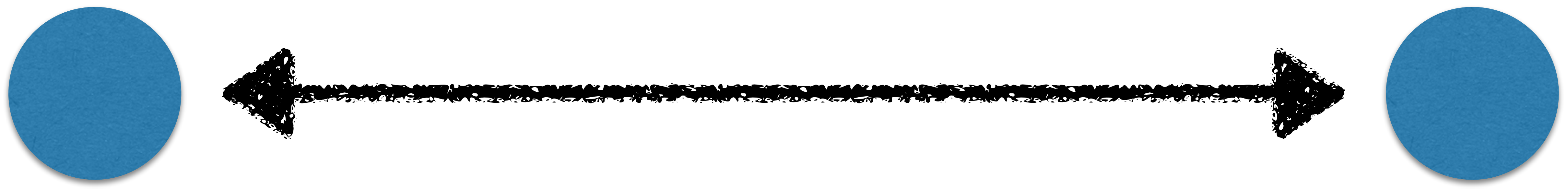
More than one way to do it

Spectrum?

Only one way to do it

assembly

`doit();`



Nothing built in

Everything built in

In the context of Rascal

- Powerful built-in features: grammars, visit, data types etc.
- But ways to “step down”

Plan

- Demonstrate grammars & concrete syntax
- Prototype a simple calculator language
- Demo Javascript language extension
- Show off Rascal :-)

- Demo

Summary

- High-level abstractions, but ways to step down
- Builtin grammars: type checking, disambiguation, optimization, concrete syntax matching
- Visit construct: structure-shy traversal

Future builtins (?)

- Coroutines
- Formatting architecture
- Faster immutable data structures (see OOPSLA'15)
- Data dependent parsing (see Onward!'15)