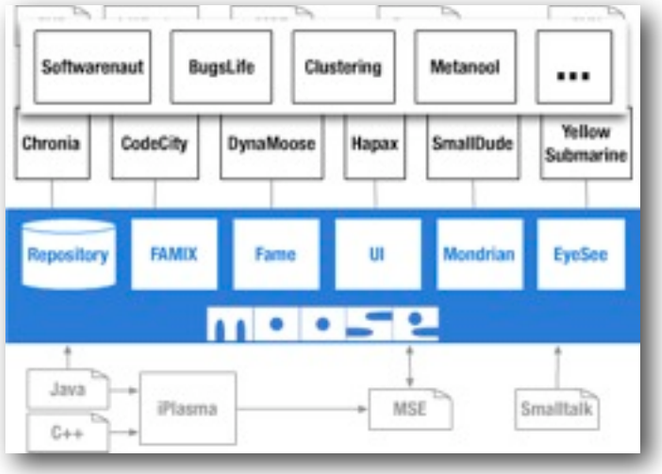
Agile Modeling

When can we have it?

Oscar Nierstrasz
Software Composition Group
scg.unibe.ch

Roadmap

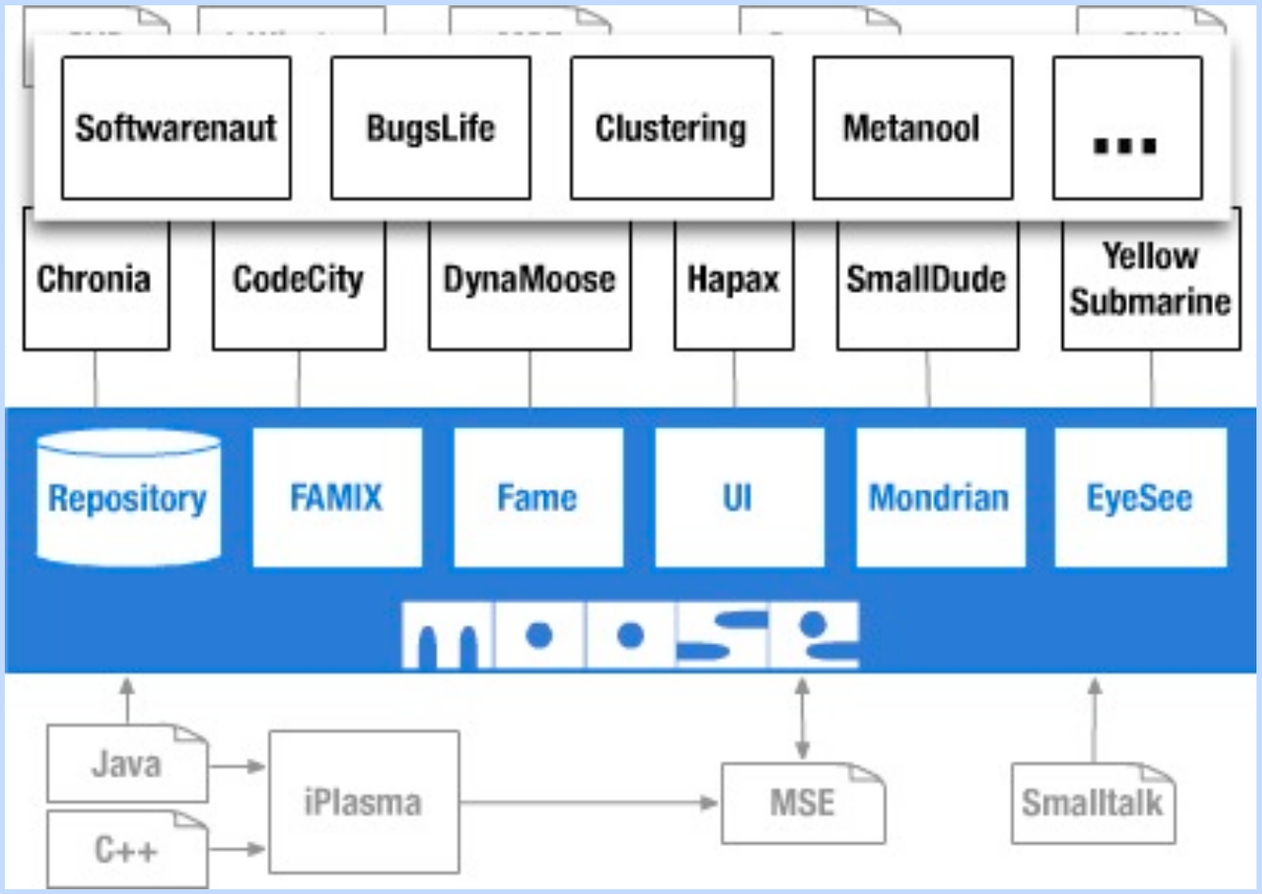


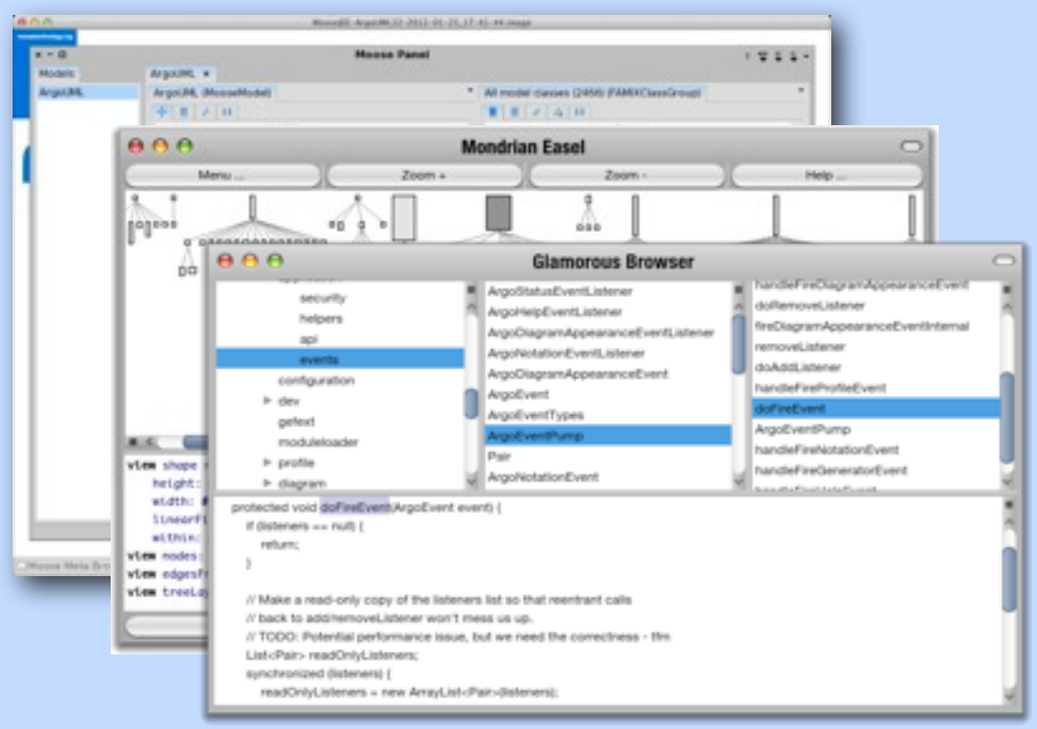
Agile Software Assessment

Agile Modeling

Directions

Agility Software Assessment





Legacy code is hard to understand

The architecture

... is not in the code

Legacy code is hard to u

not in the code

Developers spend more time reading than writing code

Specialized analyses require custom tools

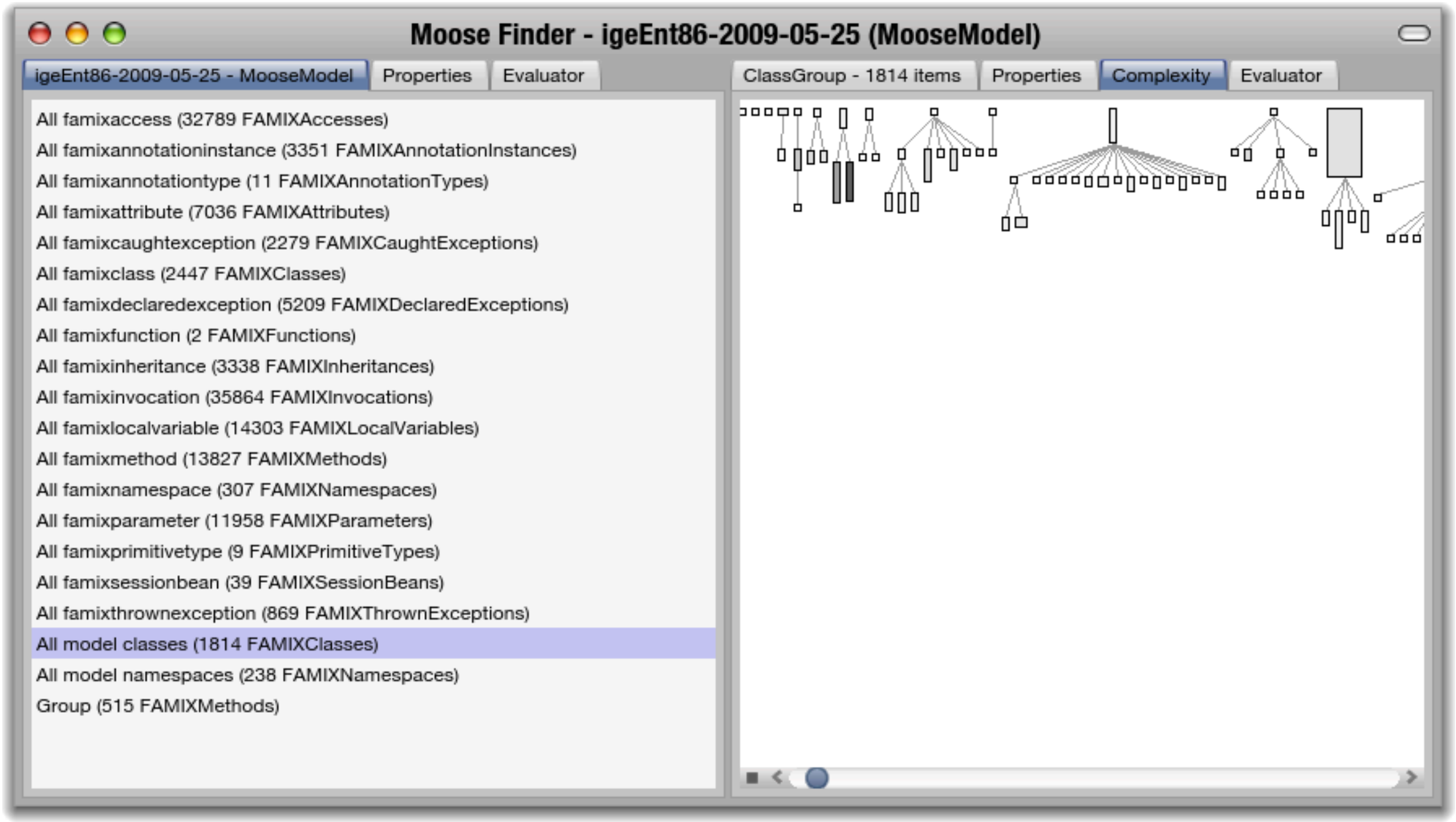
Legacy code is hard to understand

The architecture is not in the code

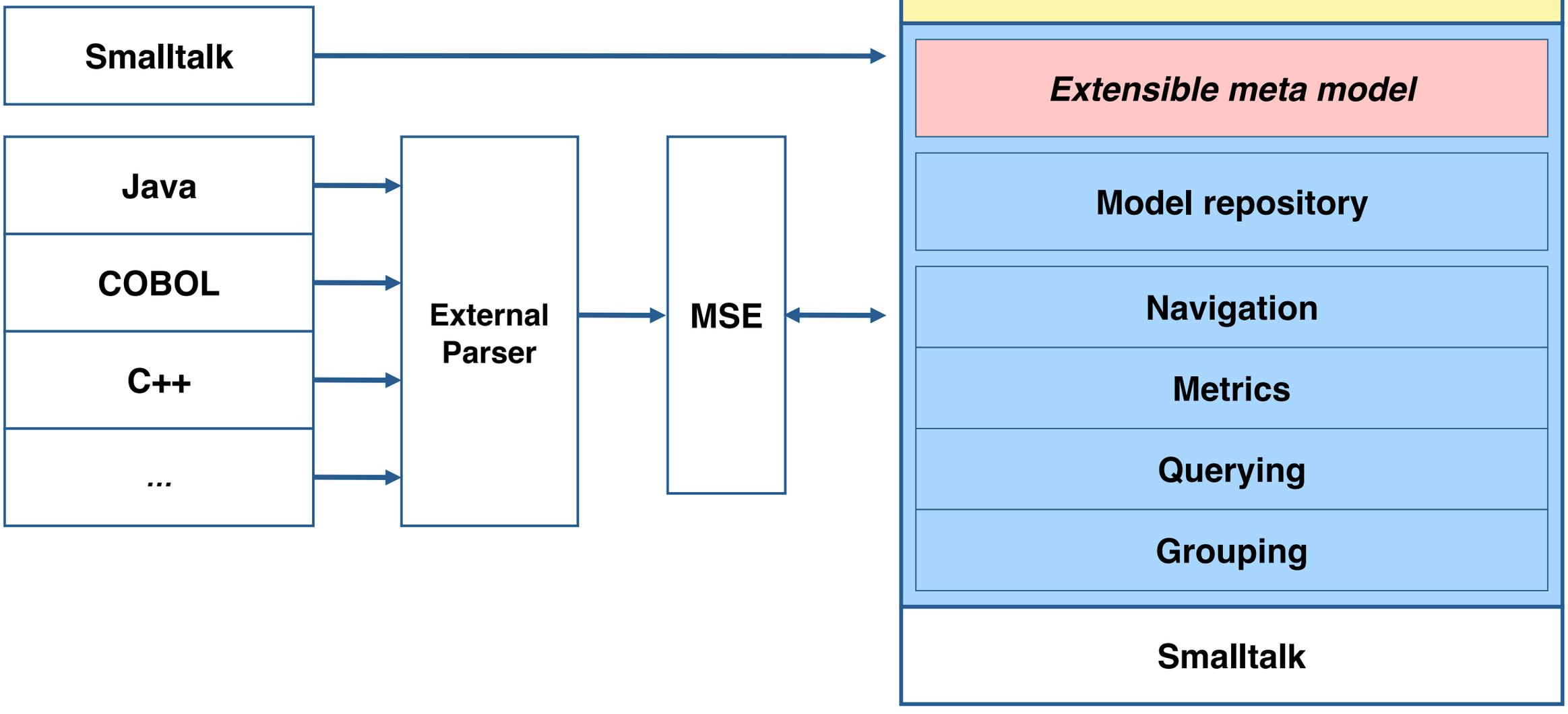
Developers spend more time reading than writing code

Specialized analyses require custom tools

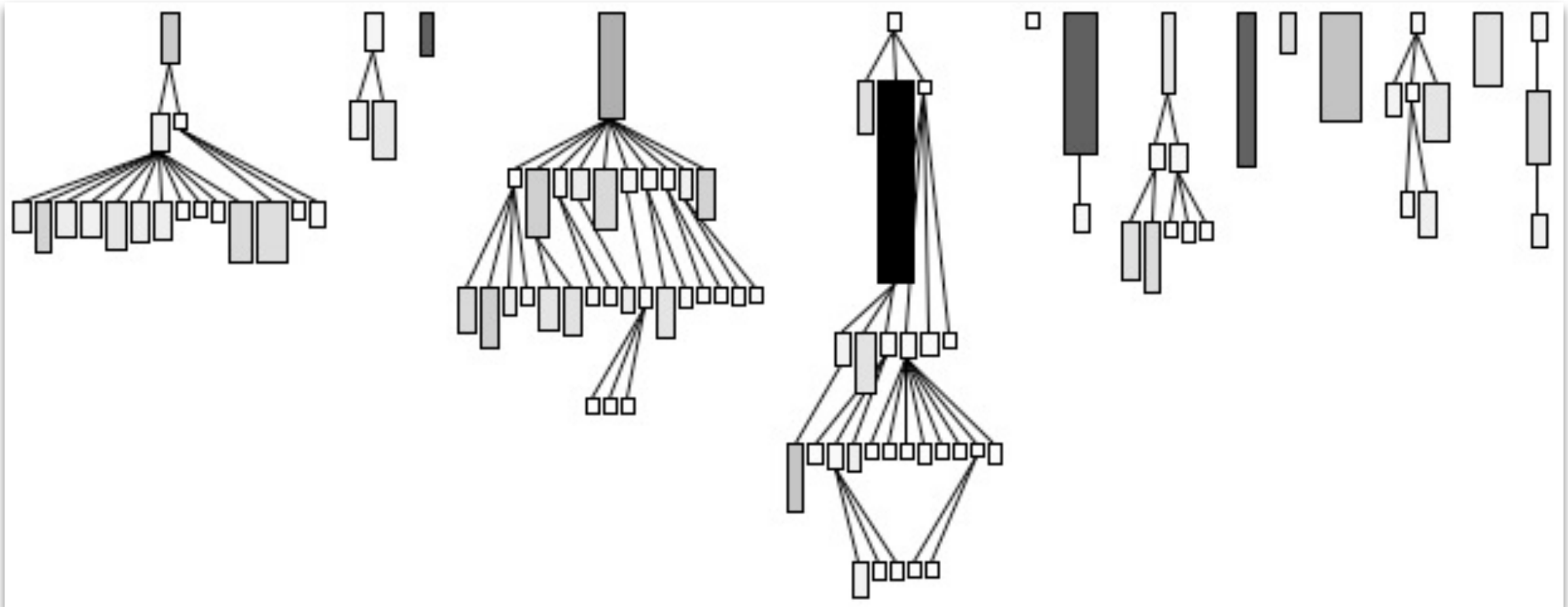
Moose is a platform for software and data analysis



www.moosetechnology.org

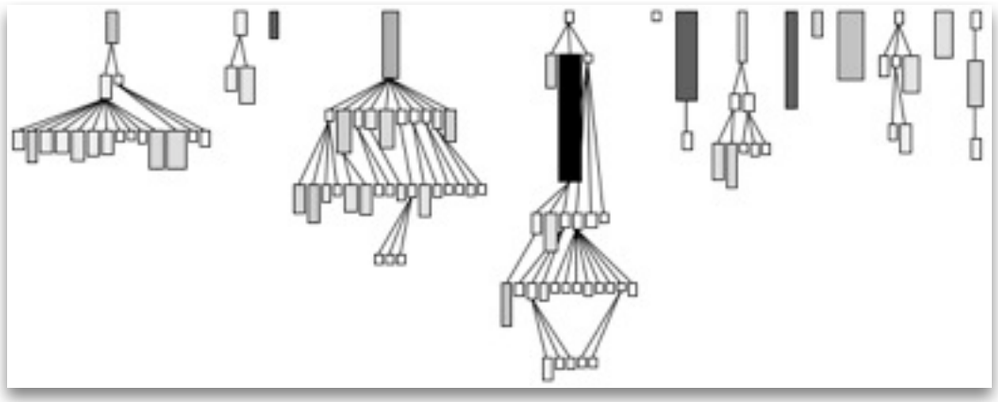


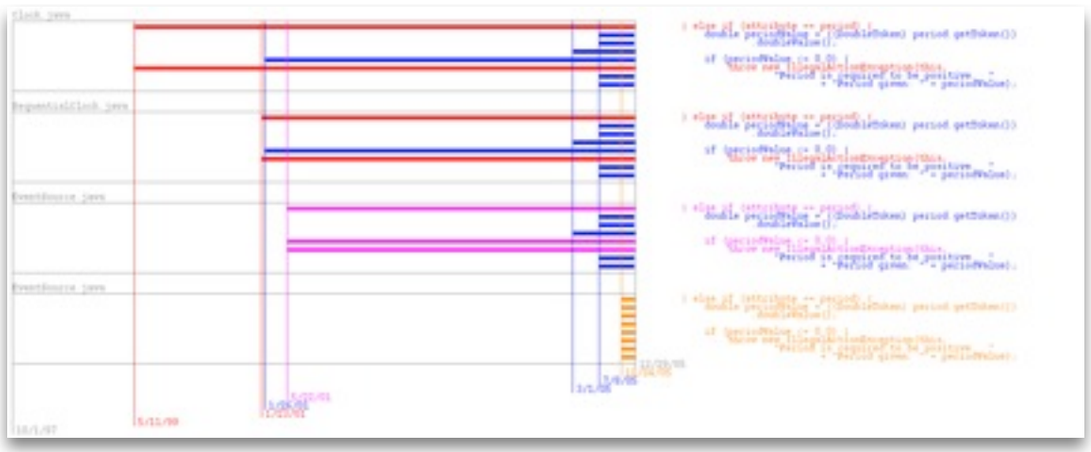
Nierstrasz et al. *The Story of Moose*. ESEC/FSE 2005

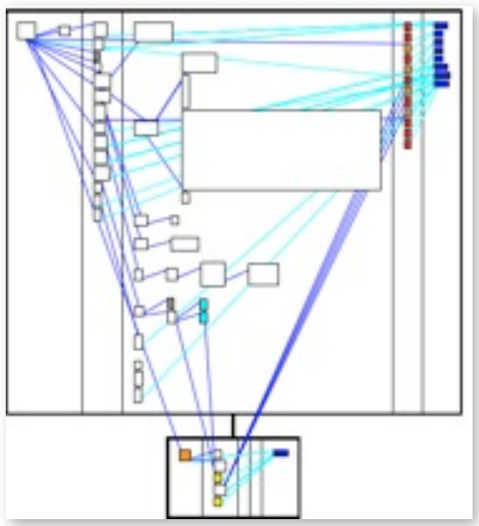


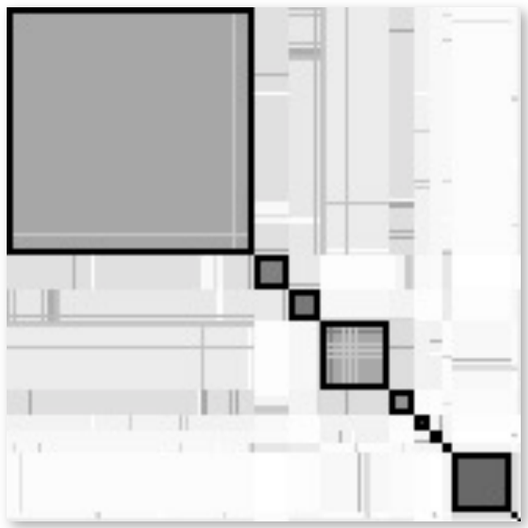
System complexity

Lanza et al. Polymetric Views. TSE 2003

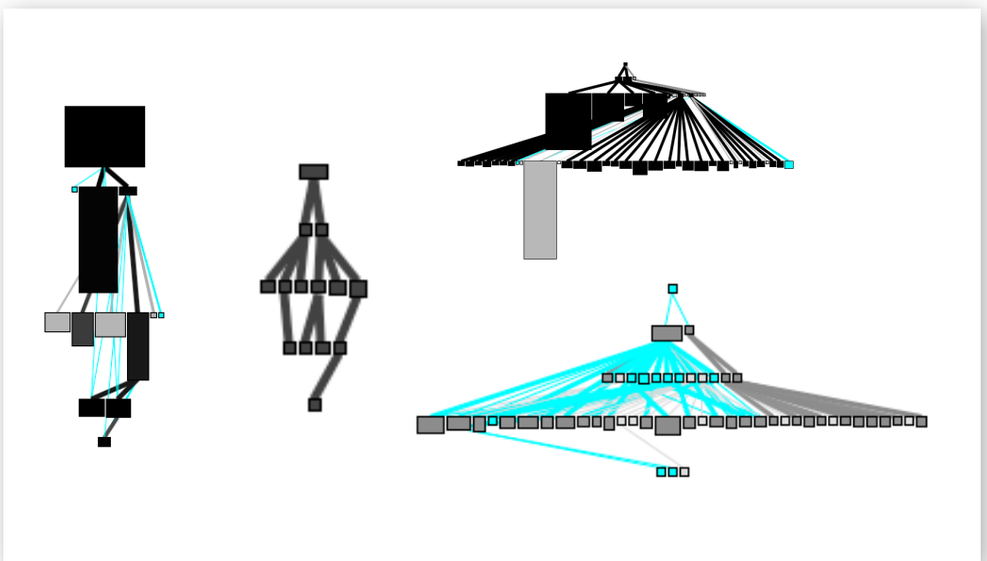


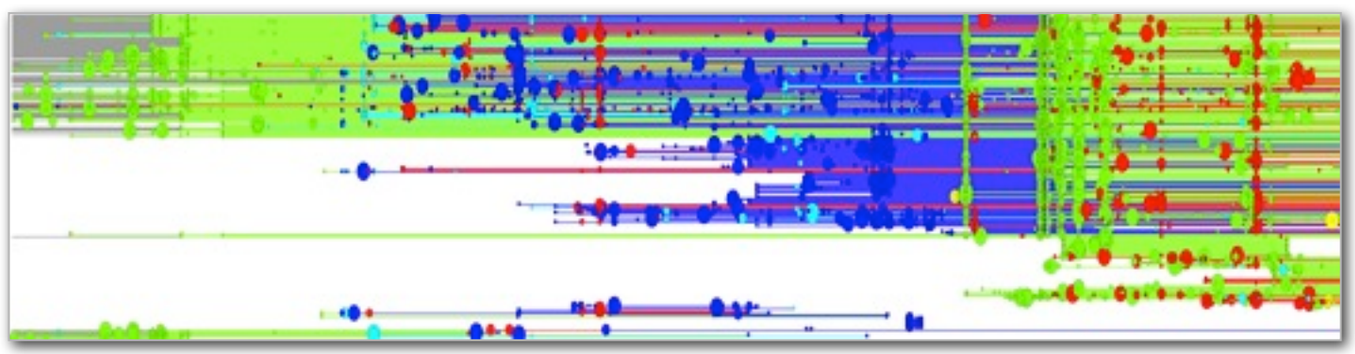







Let
com
doc
tedit
icmd
io
menu
mac
util
menu
Quickhlp
browser
macro
search
tedit
installer
options
gui
help
plugins
ref
calls
bsh
ecrip
syntax
regexp
buffer
print
netsrvs
xml
asm





Smalltalk

Java

COBOL

C++

...

ConAn | Van | Hapax | ...

CodeCrawler

Extensible meta model

Model repository

Navigation

Metrics

Querying

Grouping

Smalltalk

But, we have a huge bottleneck for new languages ...

What is Agile Software Assessment?

Challenge

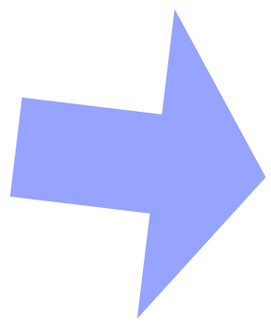
"What will my code change impact?"

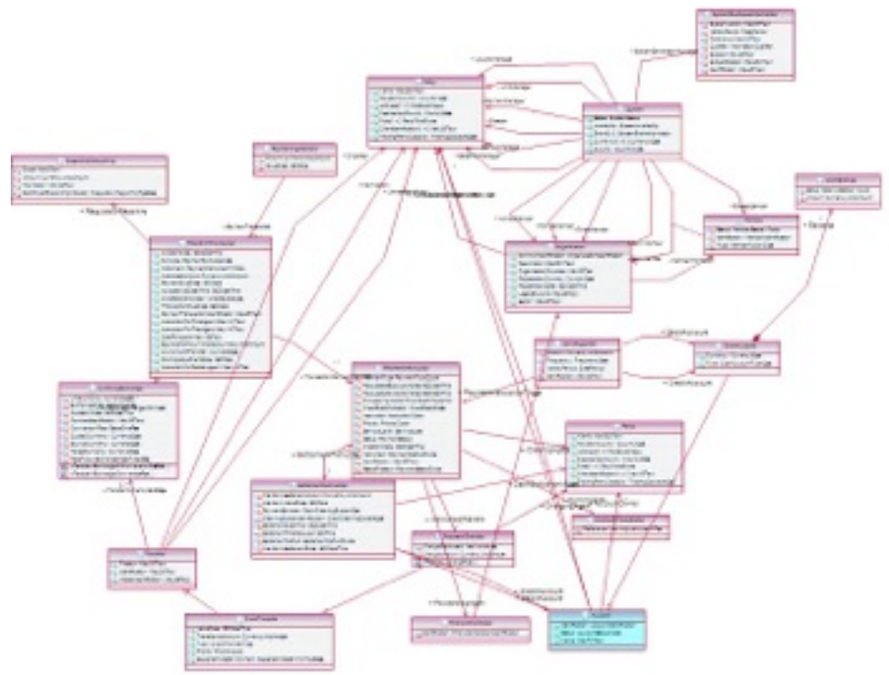
Challenge

Build a new assessment tool in ten minutes

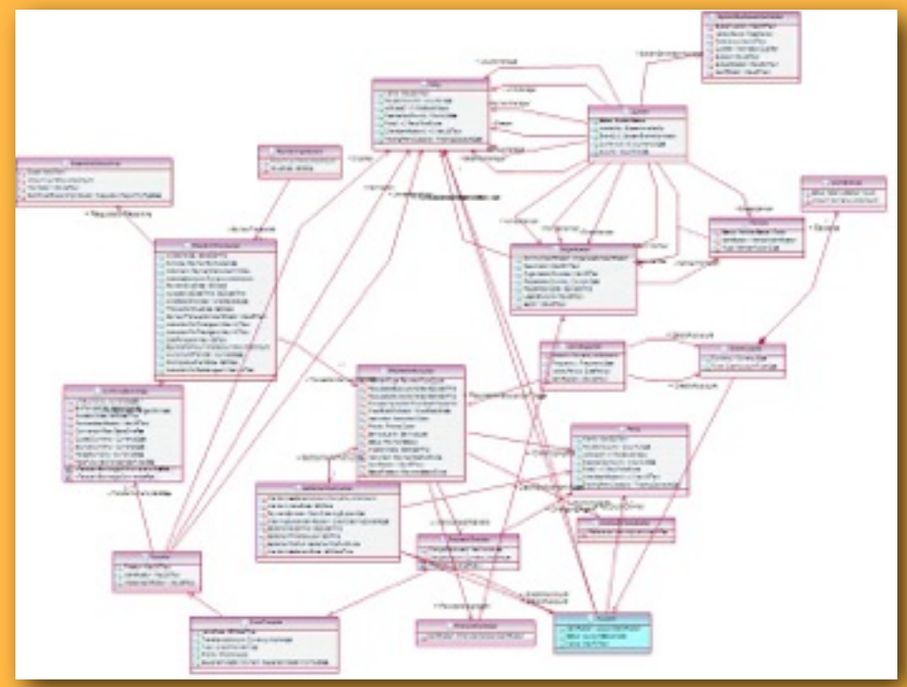
Challenge

Load the model in the morning, analyze it in the afternoon



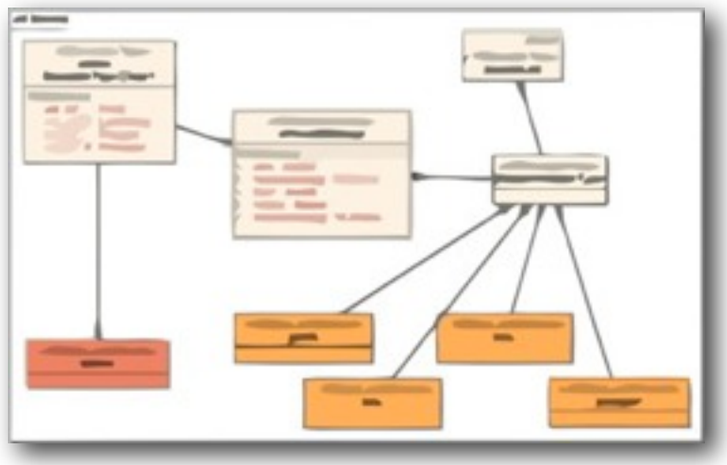


Agile Modeling



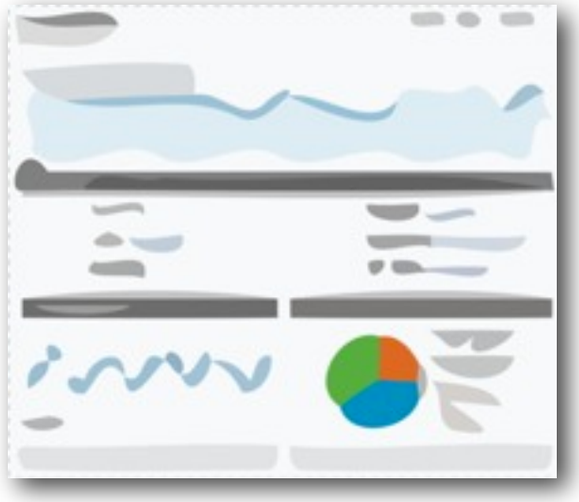
Agile Modeling Lifecycle

Build a coarse model



Refine the model

Build a custom analysis



Problems

Unknown languages

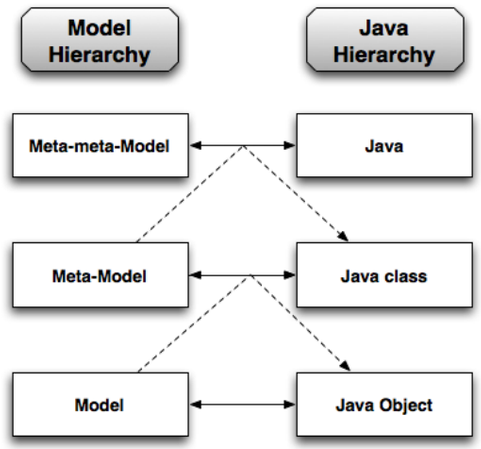
Heterogeneous projects

Unstructured text

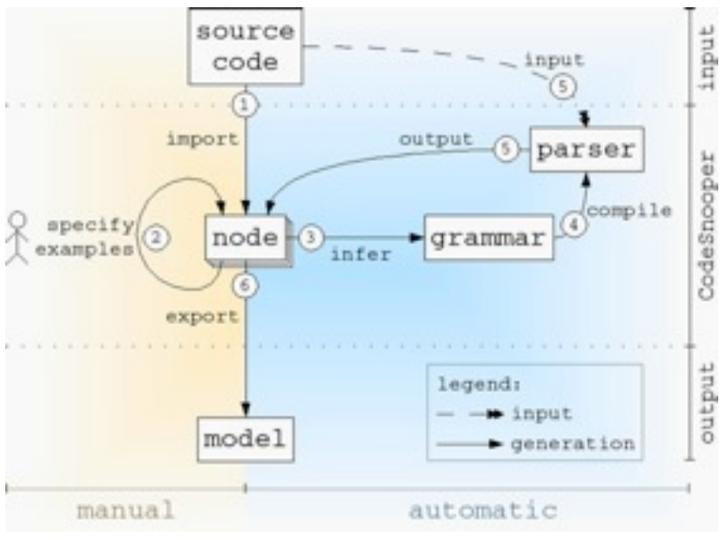
Ideas

Hooking into an existing tool

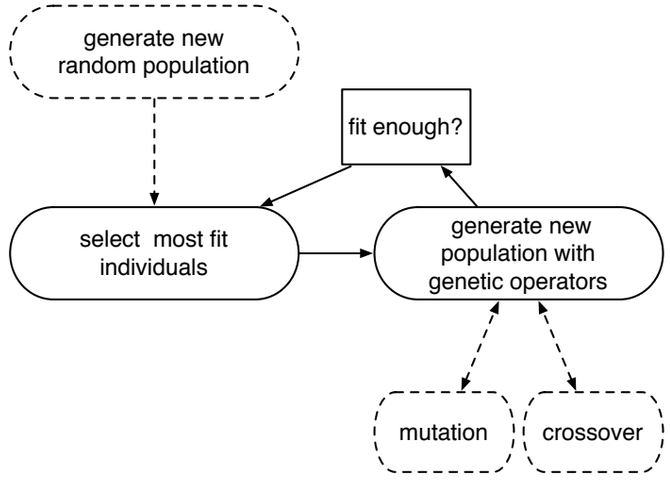
Grammar Stealing



Recycling Trees

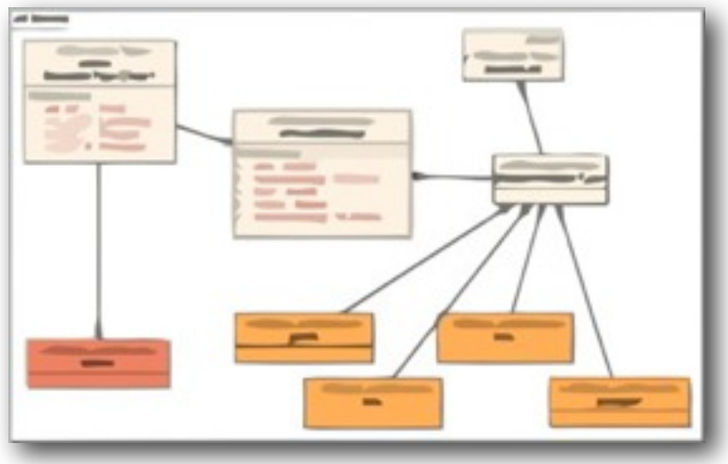


Parsing by Example



Evolutionary Grammar Generation

Hooking into an existing tool



Nice if you can have it — but just defers the problem to another platform

Grammar Stealing


feature
programming languages

Cracking the 500-Language Problem

Cracking the 500-Language Problem

Ralf Lämmel and Chris Verhoef, *Free University of Amsterdam*

Parser implementation effort dominates the construction of software renovation tools for any of the 500+ languages in use today. The authors propose a way to rapidly develop suitable parsers: by stealing the grammars. They apply this approach to two nontrivial, representative languages, PL/EX and VS Cobol II.

At least 500 programming languages and dialects are available in commercial form or in the public domain, according to Capers Jones.[1] He also estimates that corporations have developed some 200 proprietary languages for their own use. In his 1998 book on estimating Year 2000 costs, he indicated that systems written in all 700 languages would be affected.[2] His findings inspired many Y2K whistle-blowers to characterize this situation as a major impediment to solving the Y2K problem; this impediment became known as the 500-Language Problem.

In 1998, we realized that we had discovered a breakthrough in solving the 500LP—so we had something to offer regarding the Y2K problem. We immediately informed all the relevant Y2K solution providers and people concerned with the Y2K awareness campaign. In answer to our emails, we received a boilerplate email from Ed Yourdon explaining that the 500LP was a major impediment to solving the Y2K problem (which we knew, of course). Ed was apparently so good at creating awareness that this had backfired on him: he got 200 to 300 messages a day with Y2K questions and was no longer able to read, interpret, and answer his email other than in "write-only" mode. Although he presumably missed our input, his response regarding the 500LP is worth quoting:

> *I recognize that there is always a chance that someone will come up with a brilliant solution that everyone else has overlooked, but at this late date, I think it's highly unlikely. In particular, I think the chances of a "silver bullet" solution that will solve ALL y2k problems is virtually zero. If you think you have such a solution, I have two words for you: embedded systems. If that's not enough, I have three words for you: 500 programming languages. The immense variety of programming languages (yes, there really are 500!), hardware platforms, operating systems, and environmental conditions virtually eliminates any chance of a single tool, method, or technique being universally applicable.*

The number 500 should be taken poetically, like the 1,000 in the preserving process for so-called 1,000-year-old eggs, which last only 100 days. For a start, we

0740-7459/01/$10.00 © 2001 IEEE

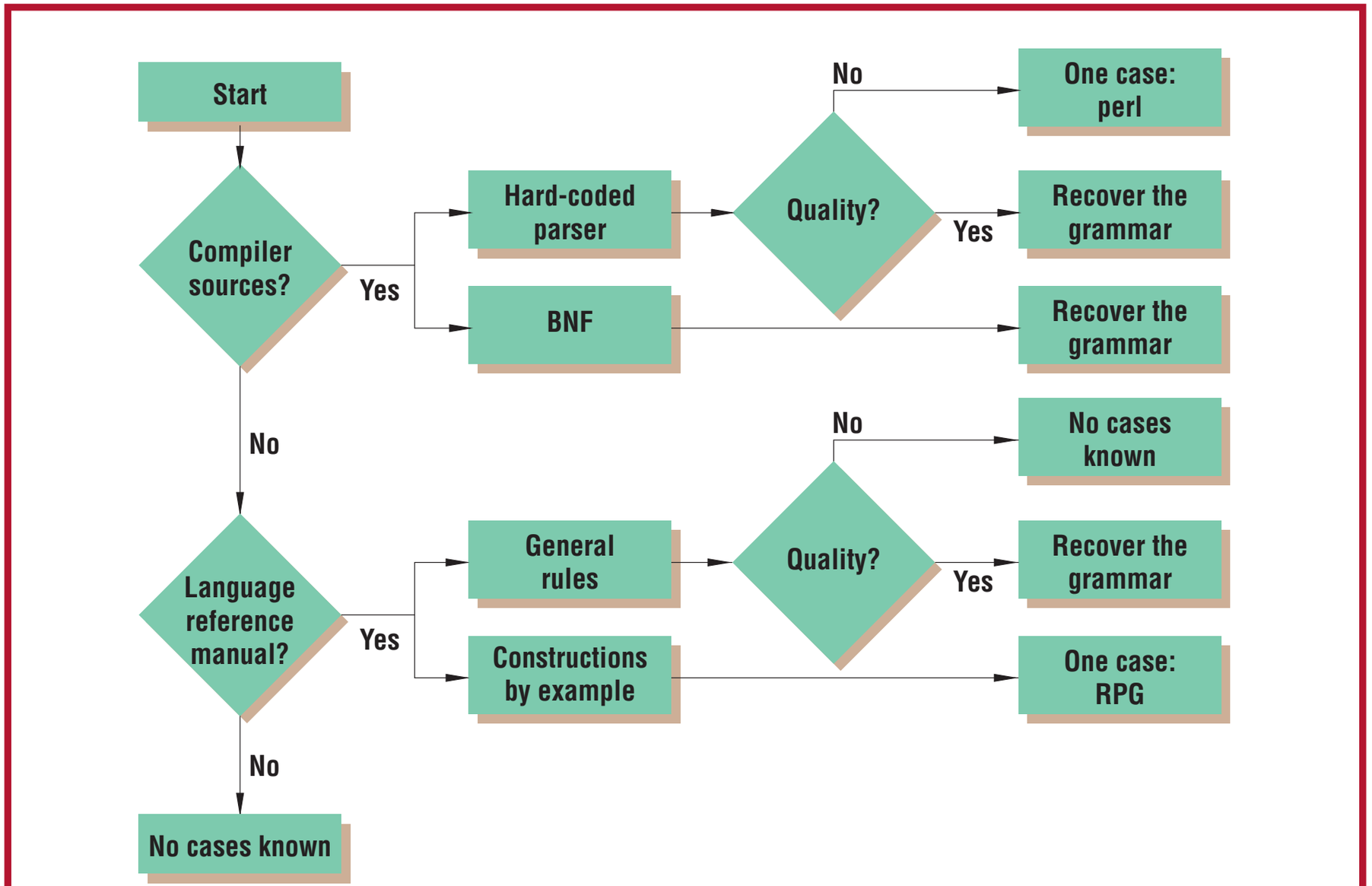
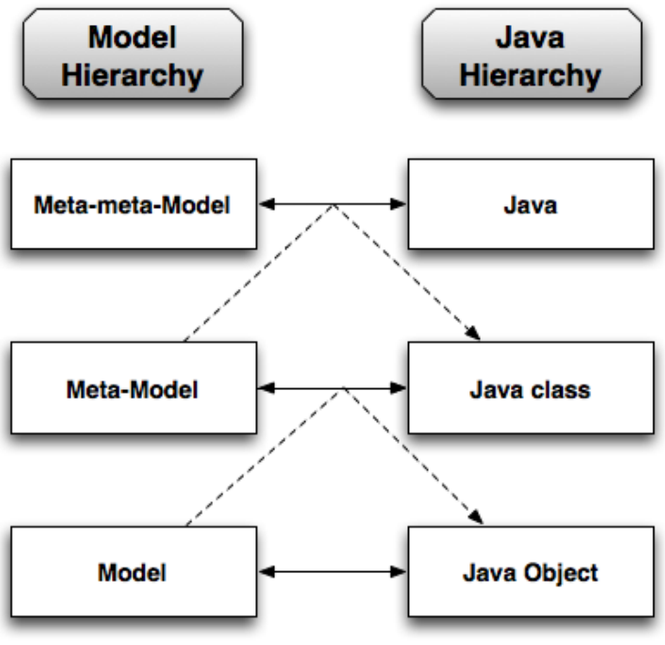


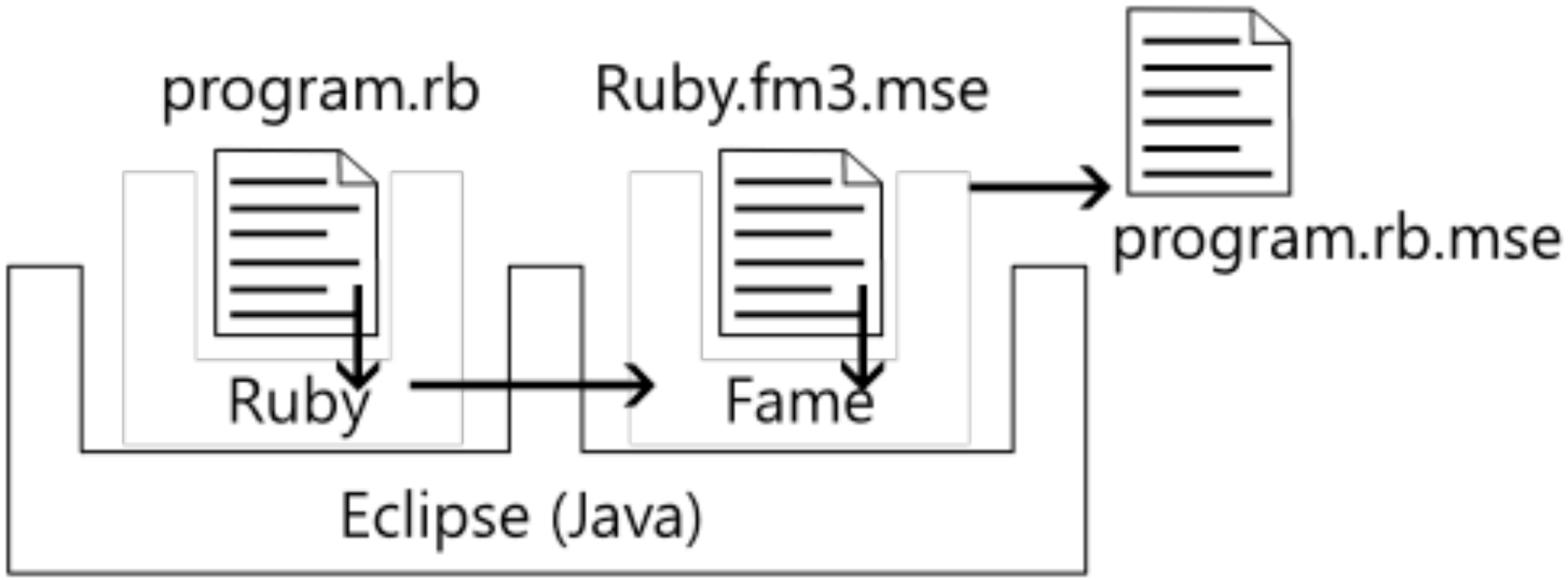
Figure 2. Coverage diagram for grammar stealing.

Still takes a couple of weeks and lots of expertise

Recycling Trees



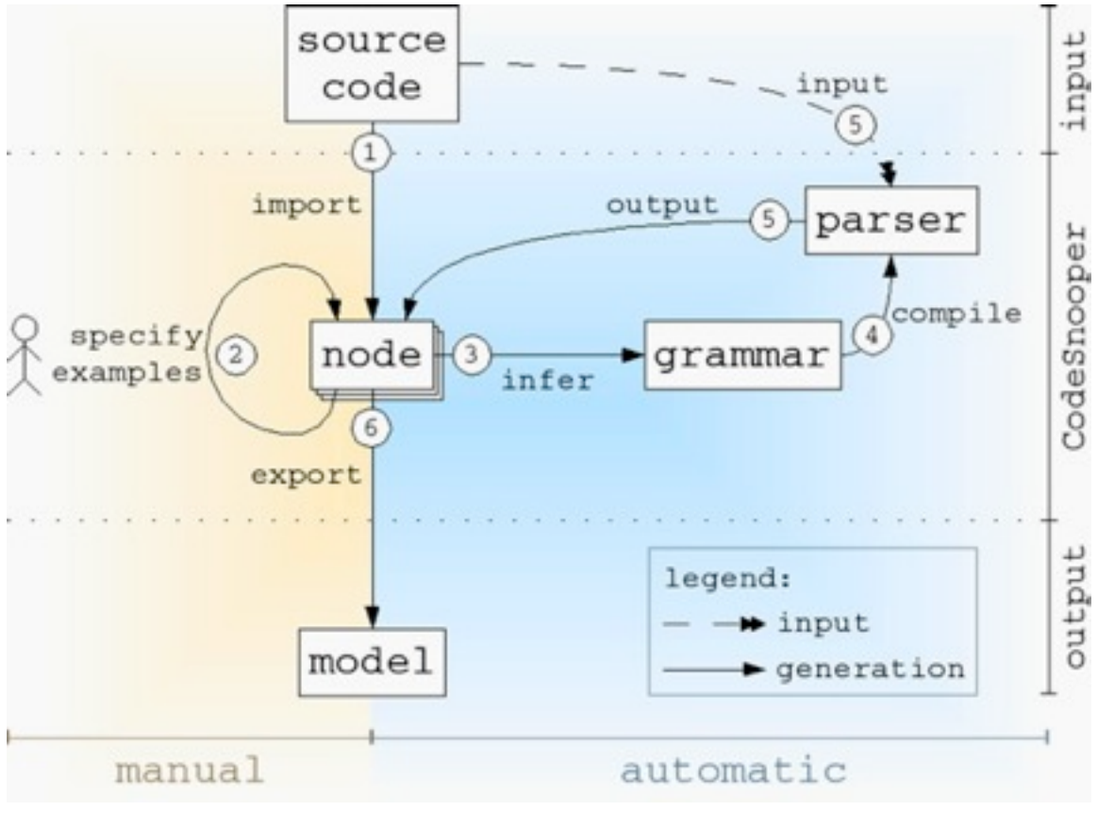
Daniel Langone. *Recycling Trees: Mapping Eclipse ASTs to Moose Models.* Bachelor's thesis, University of Bern



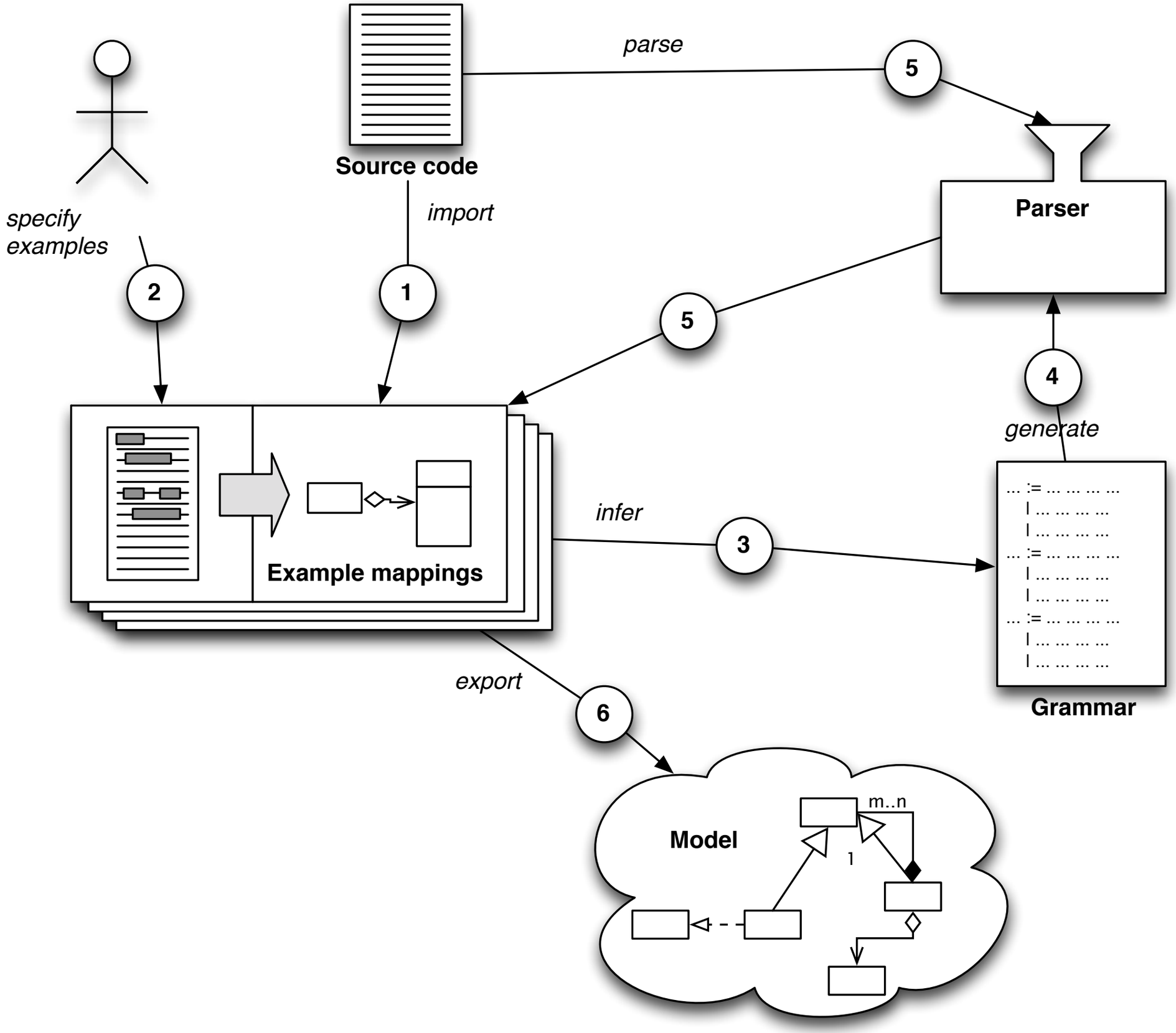
1. Infer AST implementation from IDE plugin
2. Extract metamodel from plugin
3. Map model elements to FAMIX (Moose)

Hard to recognize ASTs; still need to map to model elements

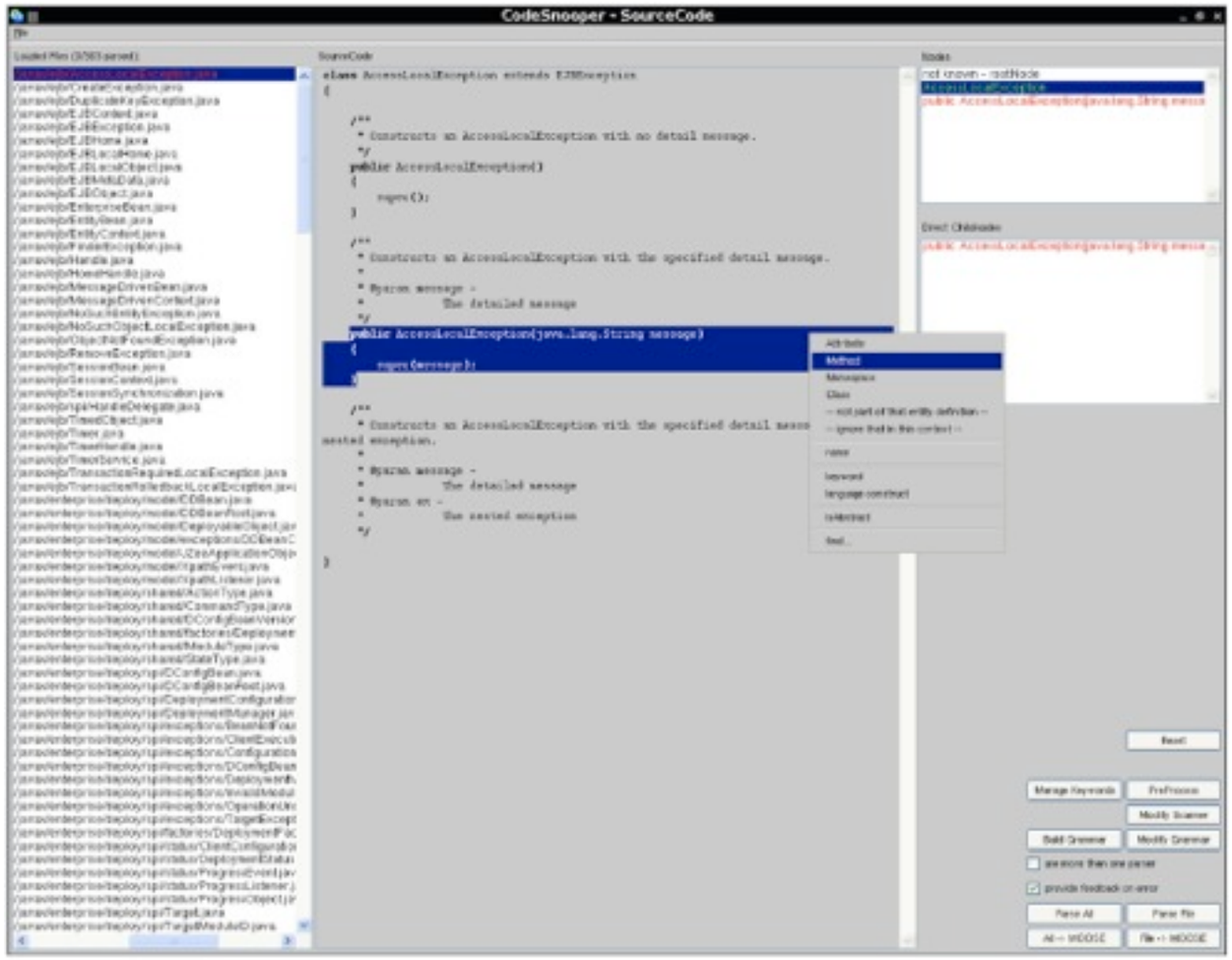
Parsing by Example



Nierstrasz et al. *Example-Driven Reconstruction of Software Models*. CSMR 2007


parse
5
Source code
Parser
specify
examples
import
2
1
5
4
generate
... := ... ...
| ... ...
| ... ...
... := ... ...
| ... ...
| ... ...
... := ... ...
| ... ...
| ... ...
infer
3
Example mappings
Grammar
export
6
Model
m..n
1

CodeSnooper



3 parsed):

SourceCode

Nodes

```
ssLocalException.java
teException.java
icateKeyException.java
Context.java
Exception.java
Home.java
ocalHome.java
ocalObject.java
MetaData.java
Object.java
priseBean.java
yBean.java
yContext.java
erException.java
lle.java
eHandle.java
sageDrivenBean.java
sageDrivenContext.java
uchEntityException.java
uchObjectLocalExceptio
ctNotFoundException.ja
oveException.java
ionBean.java
ionContext.java
ionSynchronization.java
landleDelegate.java
dObject.java
r.java
rHandle.java
rService.java
sactionRequiredLocalEx
sactionRolledbackLocal
e/deploy/model/DDBea
e/deploy/model/DDBea
e/deploy/model/Deploy
e/deploy/model/excepti
e/deploy/model/J2eeAp
e/deploy/model/XpathE
e/deploy/model/XpathL
e/deploy/shared/Action
e/deploy/shared/Comma
e/deploy/shared/DConf
e/deploy/shared/factor
e/deploy/shared/Modul
e/deploy/shared/StateT
e/deploy/spi/DConfigBean.java
e/deploy/spi/DConfigBeanRoot.java
e/deploy/spi/DeploymentConfiguratior
```

```
class AccessLocalException extends EJBException
{
    /**
     * Constructs an AccessLocalException with the specified
     *
     * @param message -
     *            The detailed message
     */
    public AccessLocalException(java.lang.String message)
    {
        super(message);
    }

    /**
     * Constructs an Acce                                    cified
nested exception.
     *
     * @param message -
     *            The det
     * @param ex -
     *            The nes
     */

}
```

```
not known - rootN
AccessLocalExce
public AccessLoc
```

Direct Childnodes

```
public AccessLoc
```

Attribute

Method

Namespace

Class

-- not part of that entity definition --

-- ignore that in this context --

name

keyword

language construct

isAbstract

find...

hat entity definition --

this context --

ruct

	Precise Model	Our Model
Number of Model Classes	366	346
Number of Abstract Classes	233	230
Total Number Of Methods	1887	1780
Total Number of Attributes	395	304

JBoss case

	Precise Model	7 files	Our Model
Number of Namespaces	8	6	6
Number of Model Classes	25	4	4
Total Number of Methods	247	26	26
Total Number of Attributes	136	9	9

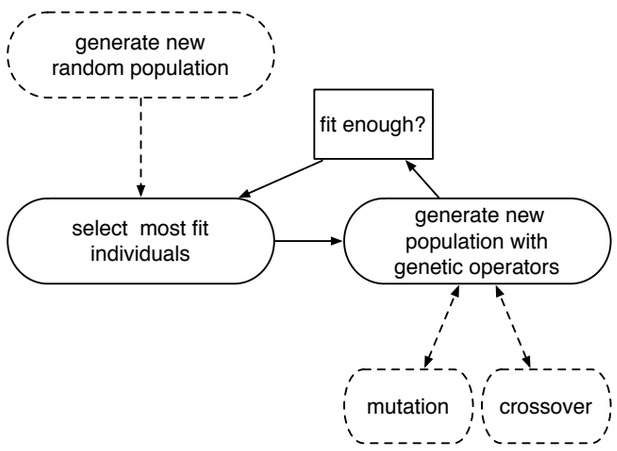
Ruby case

Problems

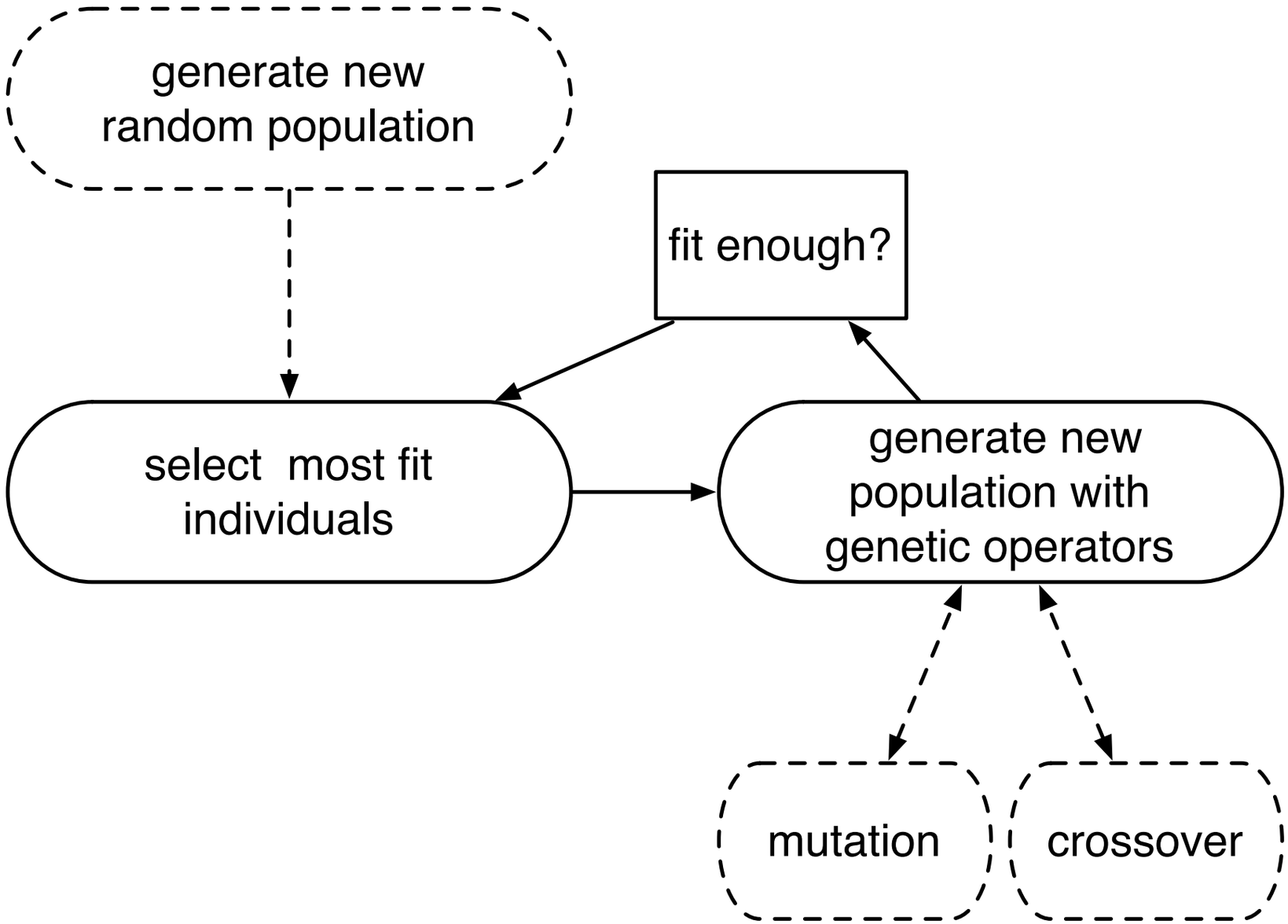
- **Ambiguity**
- **False positives**
- **False negatives**
- **Embedded languages**

Markus Kobel. *Parsing by Example*. MSc, U Bern, April 2005.

Evolutionary Grammar Generation



Sandro De Zanet. *Grammar Generation with Genetic Programming — Evolutionary Grammar Generation.* MSc, U Bern, July 2009.


generate new
random population
fit enough?
select most fit
individuals
generate new
population with
genetic operators
mutation
crossover

PEG mutation and crossover

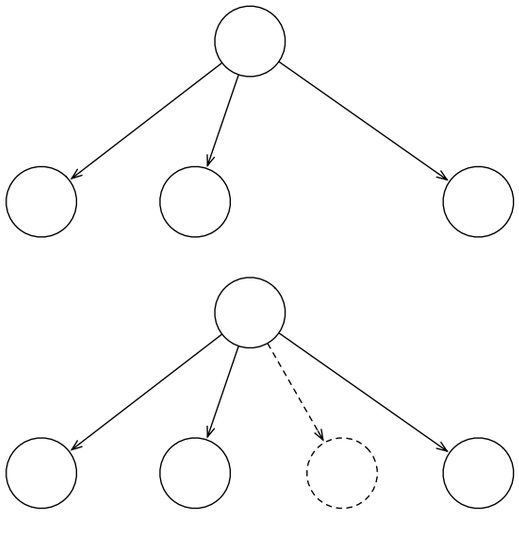


Figure 4.1: Add a node

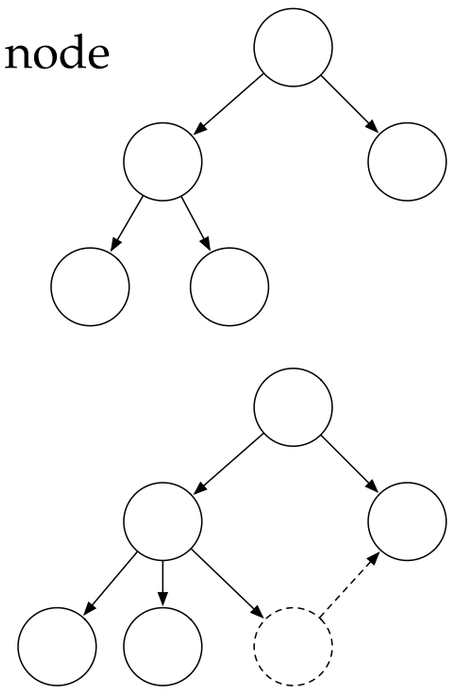


Figure 4.2: Add back link node

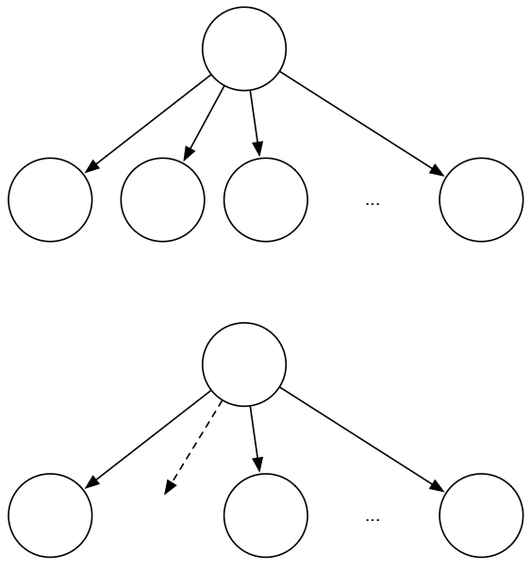


Figure 4.3: Delete a node

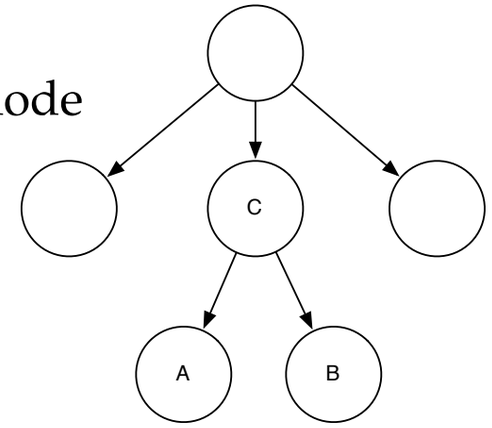


Figure 4.4: Push a node up

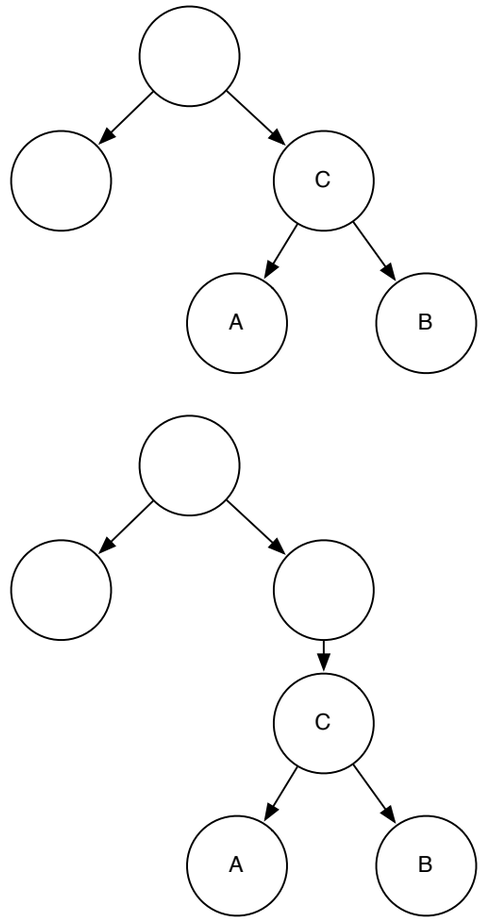


Figure 4.5: Insert a node

Desired grammar:

```
([a-z] (‘_‘ | [0-9] | [a-z])*)
```

Found grammar:

```
(([a-z] ({‘\n‘ | ‘_‘ | [0-9]})*))*
```

Slow and expensive. Modest results for complex languages.

Desired grammar:

```
0 -> (‘c‘ ‘a‘ ‘t‘ ‘:‘ ‘ ‘ ([a-z])+ 1 -> {2 -> (‘\n‘ 0) | e})
```

Found grammar:

```
(0 -> (‘c‘ ‘a‘ ‘t‘ ‘:‘ ‘ ‘ 2 -> (([a-z])+ ‘\n‘)))+
```

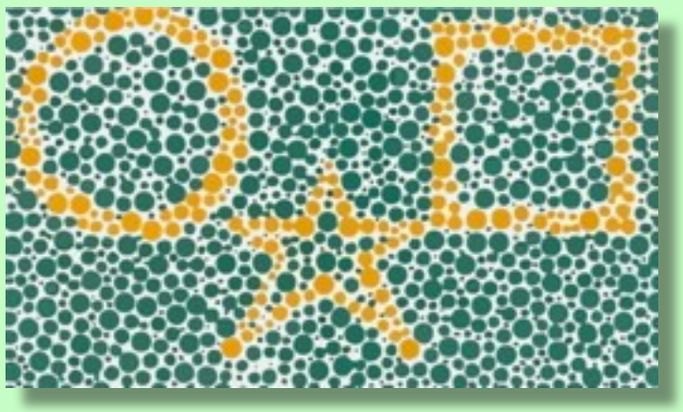
Desired grammar:

```
0 -> (‘+‘ | ‘-‘ | ‘<‘ | ‘>‘ | ‘,‘ | ‘.‘ | 1
-> (‘[‘ 2 -> (0)* ‘]‘))
```

Found grammar:

```
(0 -> {‘<‘ | ‘]‘ | ‘.‘ | ‘,‘ | ‘>‘ | ‘-‘ | ‘[‘ | ‘+‘})*
```

Directions



Incrementally refine island grammars

Progress

Global Islands are useful
Scoped Islands are useful

Recursive Islands and Scoped Repeating Islands lead to Left-Recursion Problems

Verification of a scope is expensive

Tokenization and memoization help :-)

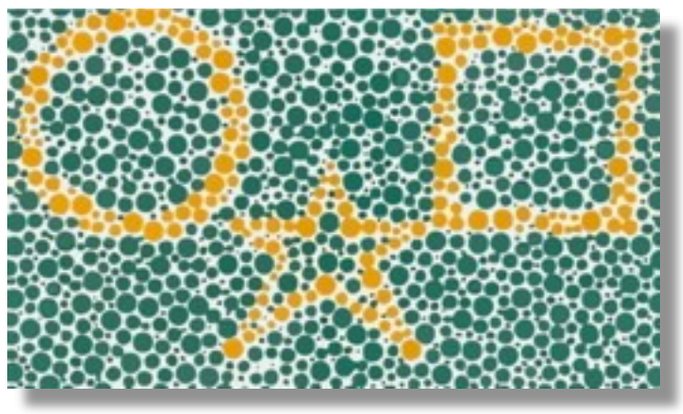
Composing parsers from parts

Ideas

Exploit similarities between languages (adapt and compose)

- **similar syntax, similar constructs**
- **combine “reusable” islands?**
- **adapt with genetic algorithms?**
- **combine with parsing by example?**

Automatic structure recognition



Ideas

Exploit indentation as a proxy for structure

```
catch, throw, try, bool, class, enum,
explicit, export, friend, inline, mutable,
namespace, operator, private, protected,
public, template, typename, using,
virtual, volatile, wchar_t, and, and_eq,
bitand, bitor, compl, const_cast, delete,
dynamic_cast, false, new, not, not_eq, or,
or_eq, reinterpret_cast, static_cast,
this, true, typeid, xor, xor_eq
```

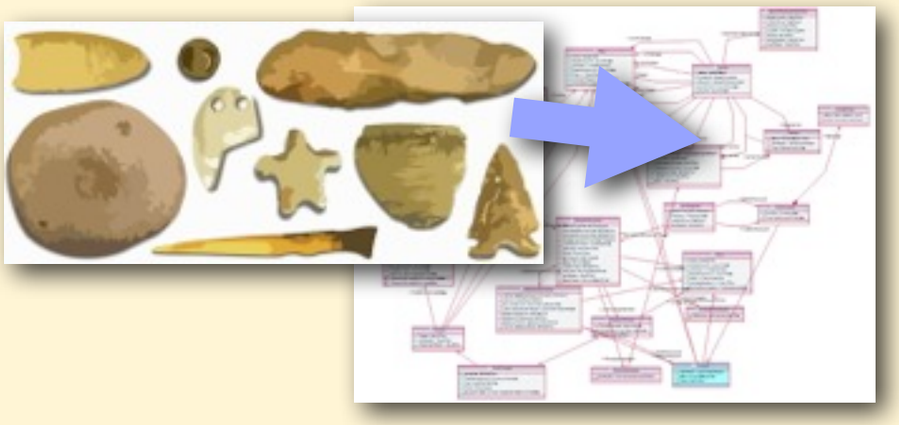
Heuristics to automatically detect keywords

- **don’t parse; just recognize structure?**
- **combine with parsing by example?**
- **combine with grammar evolution?**
- **combine with parser composition?**

Conclusions

Model construction is an obstacle to agile software assessment

You don't need precise parsers to start analysis



Are there effective shortcuts to building a parser/importer by hand?