Domain-Specific Tooling

Oscar Nierstrasz
Software Composition Group
scg.unibe.ch

u^b
UNIVERSITÄT BERN

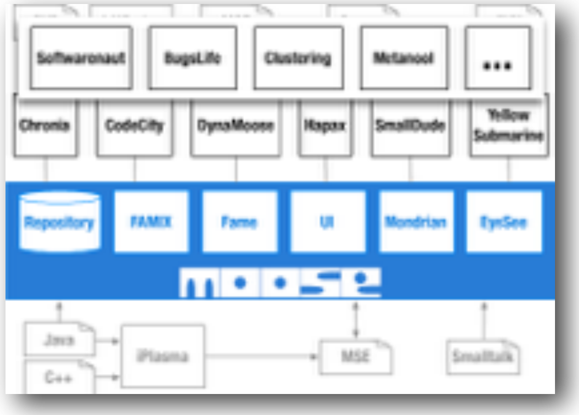
A
A S A
A

Roadmap

Agile Software Assessment

Agile Modeling

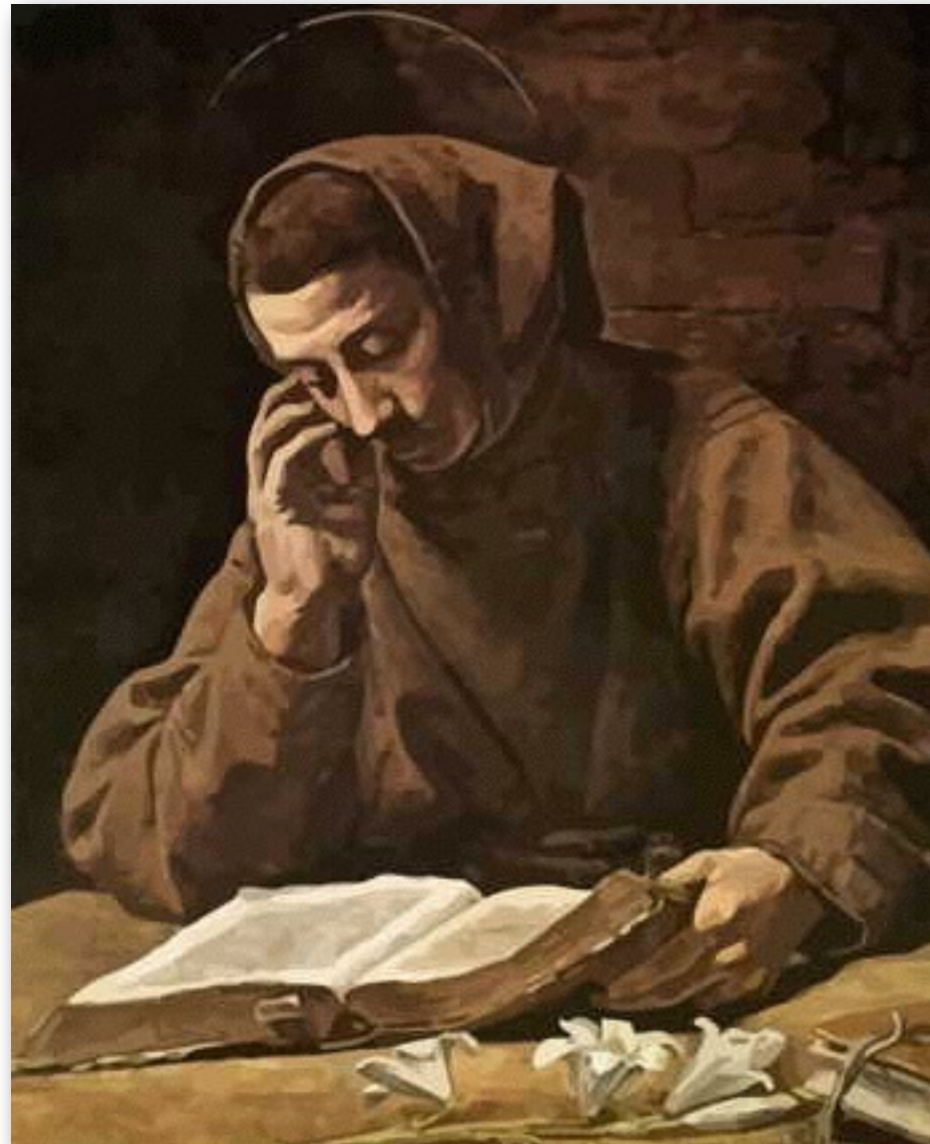
Architectural Monitoring



Agility in Moose

Moldable Tools

Agile Software Assessment



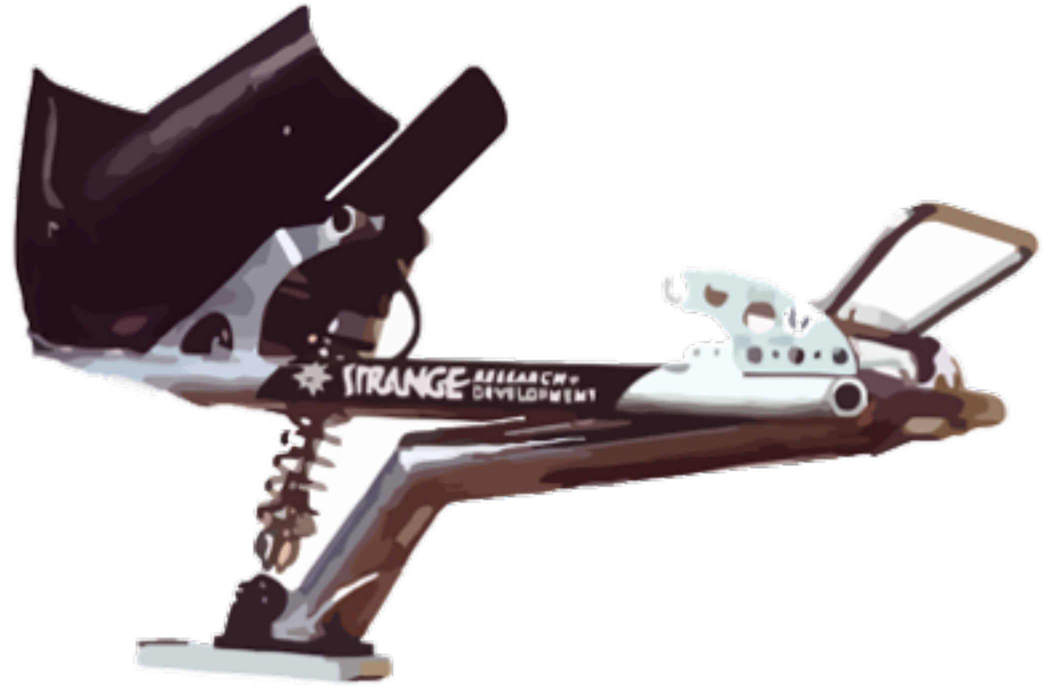
Developers spend more time reading than writing code

There is a gap between Models

and Code

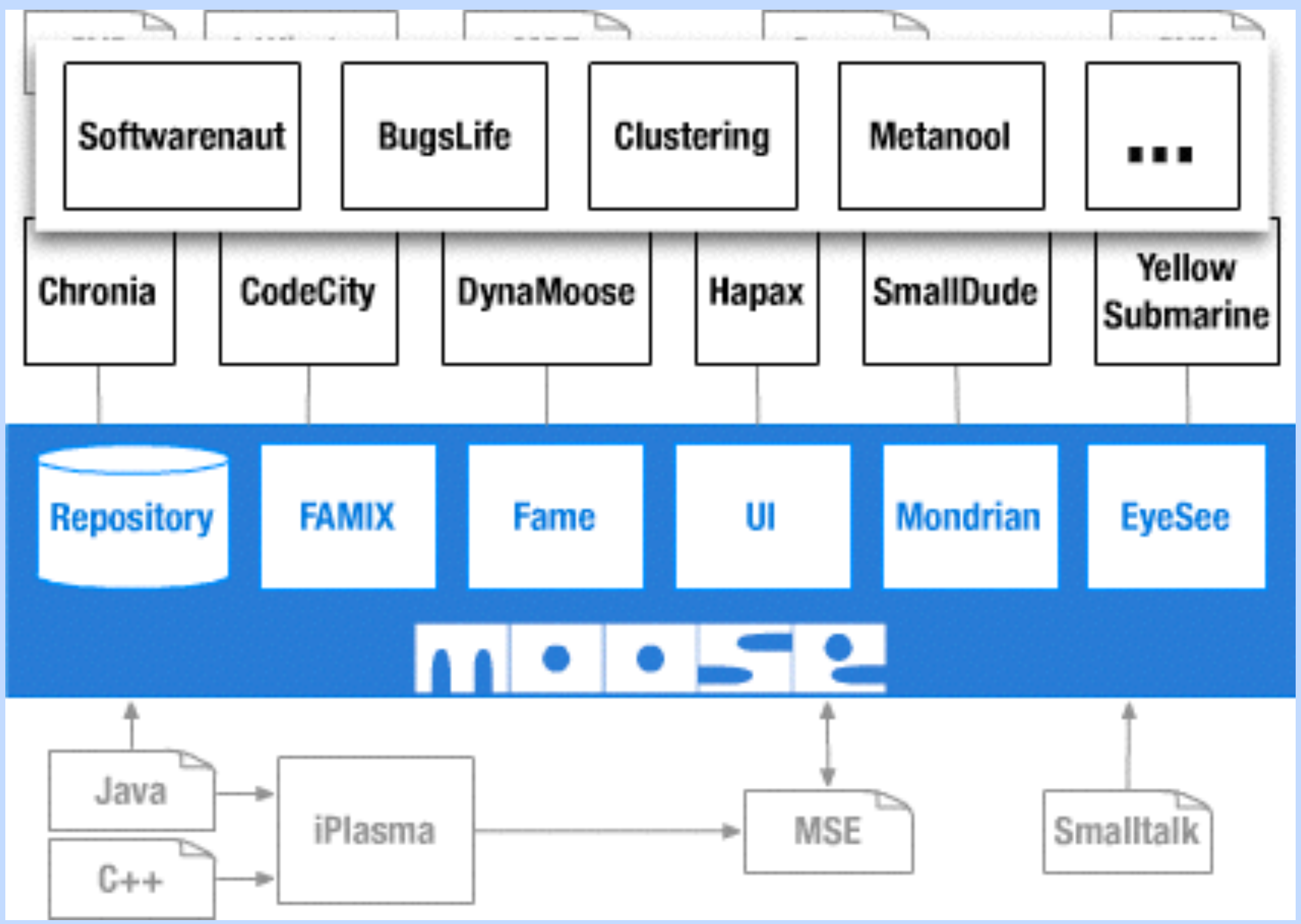
The architecture

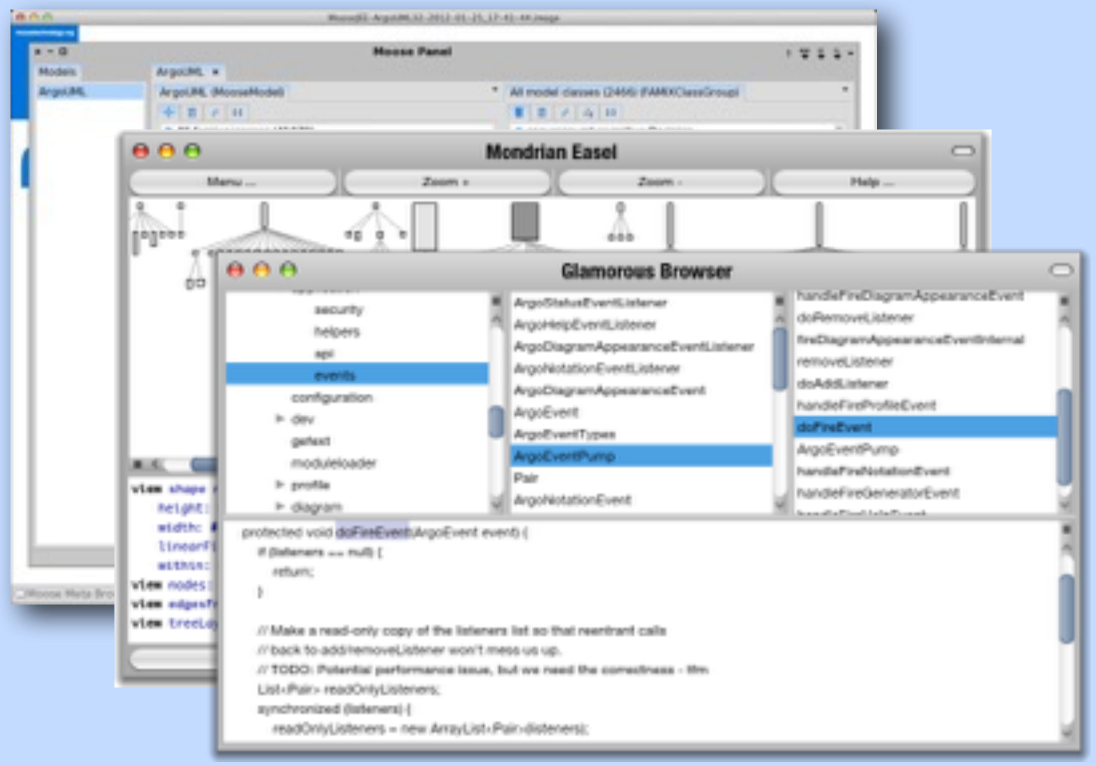
... is not in the code



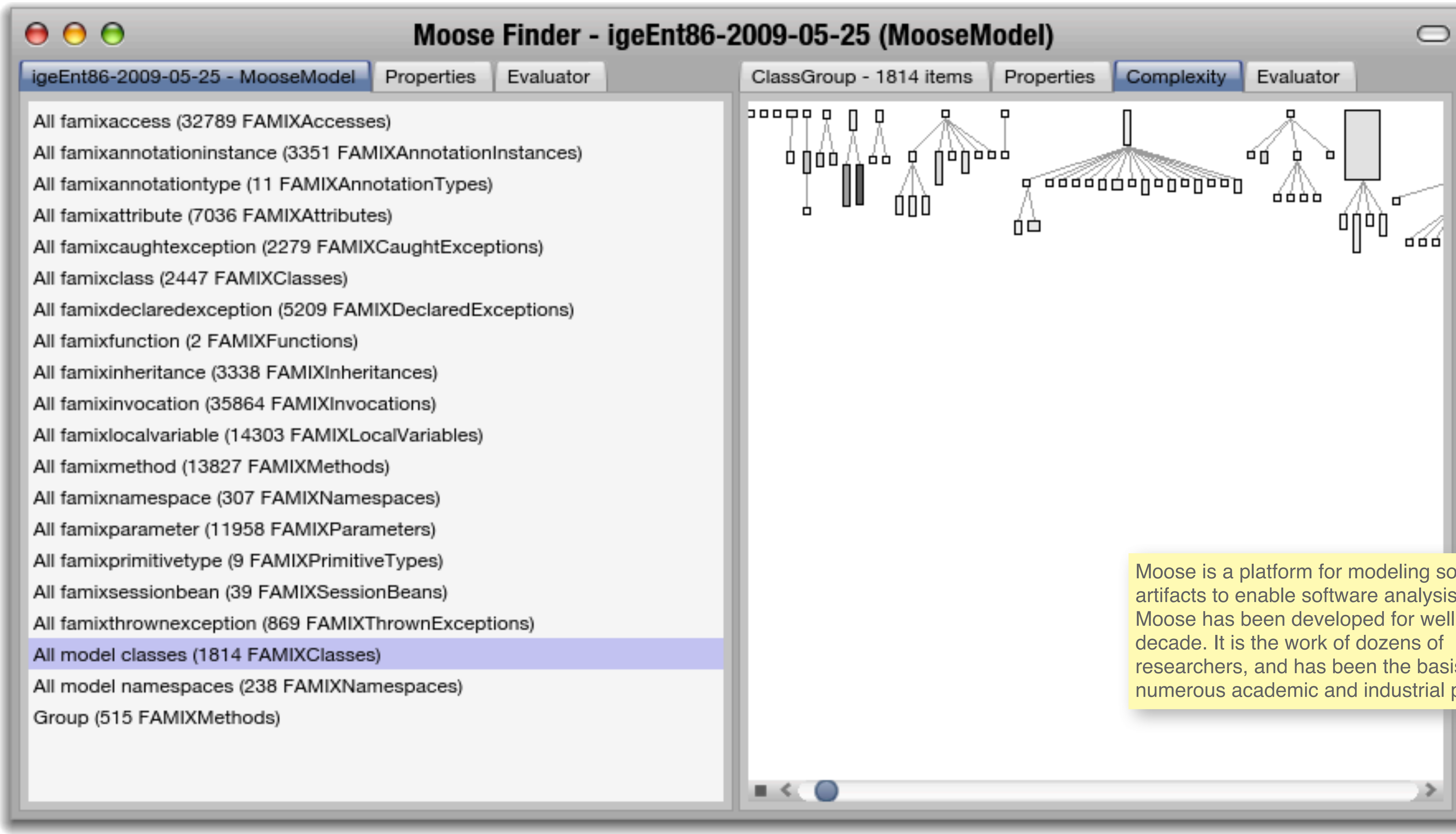
Specialized analyses require custom tools

Agility in Moose



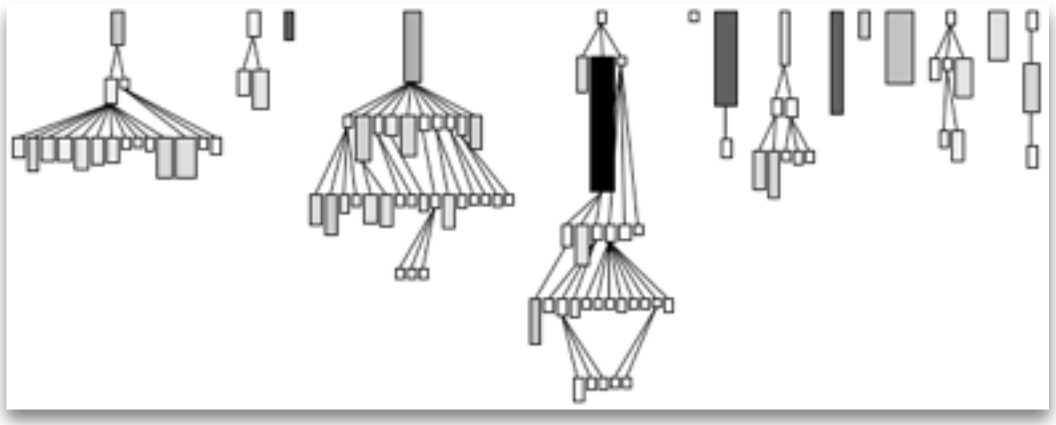


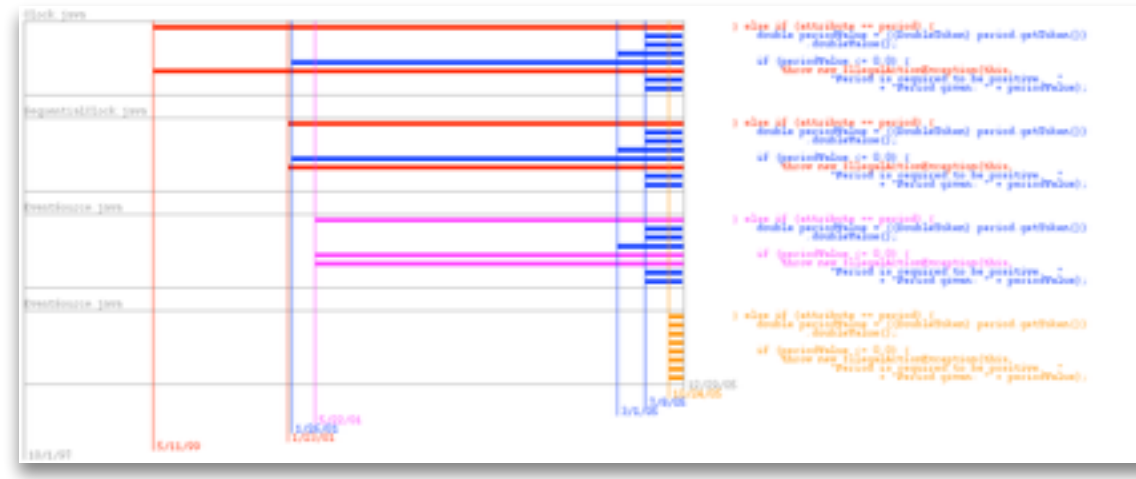
Moose is a platform for software and data analysis

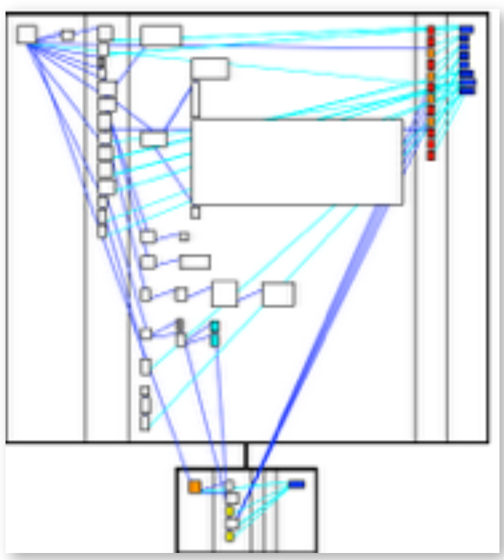


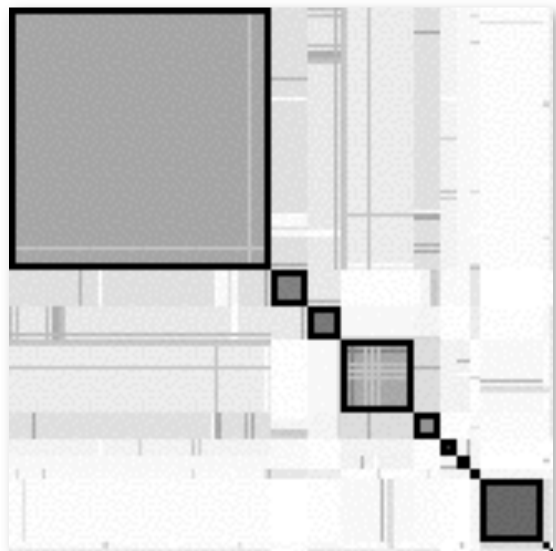
Moose is a platform for modeling software artifacts to enable software analysis. Moose has been developed for well over a decade. It is the work of dozens of researchers, and has been the basis of numerous academic and industrial projects.

www.moosetechnology.org

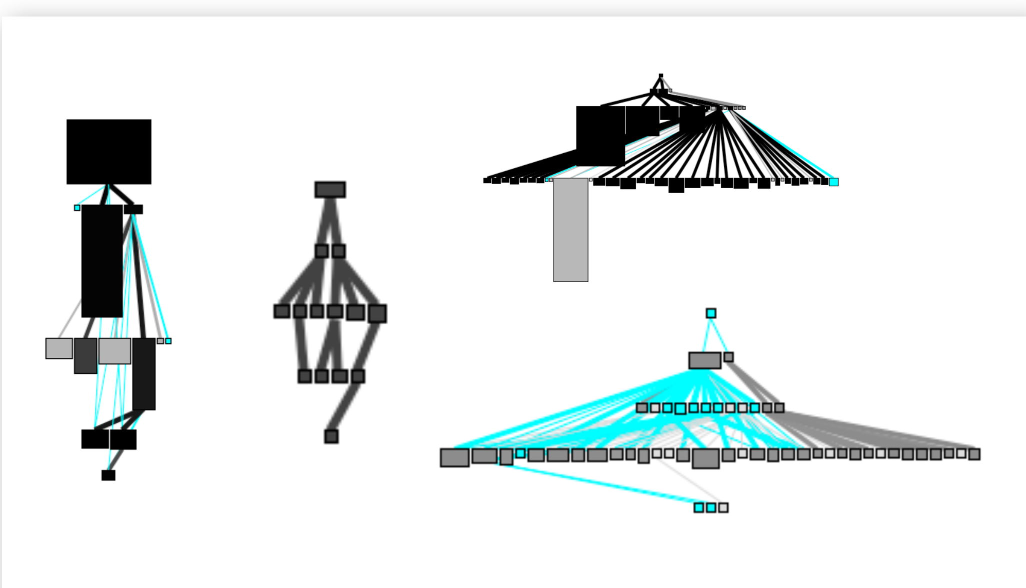


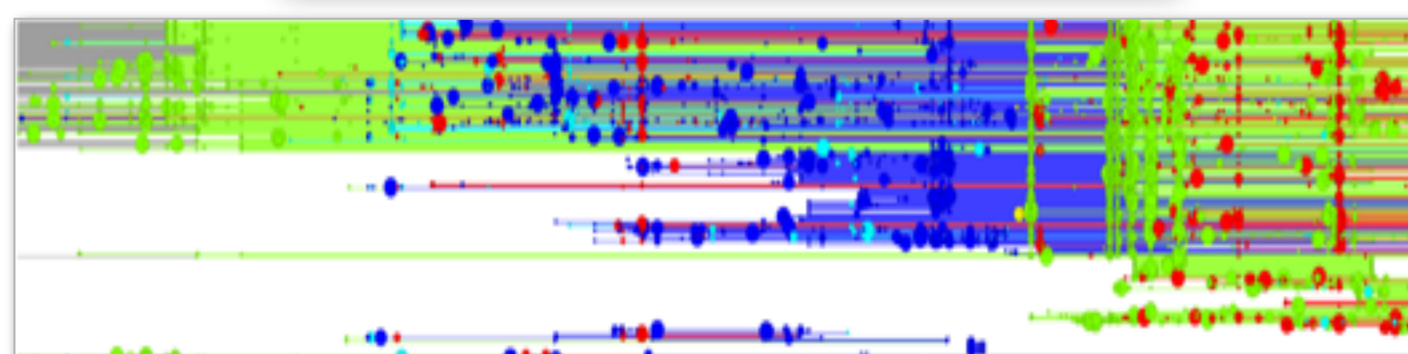




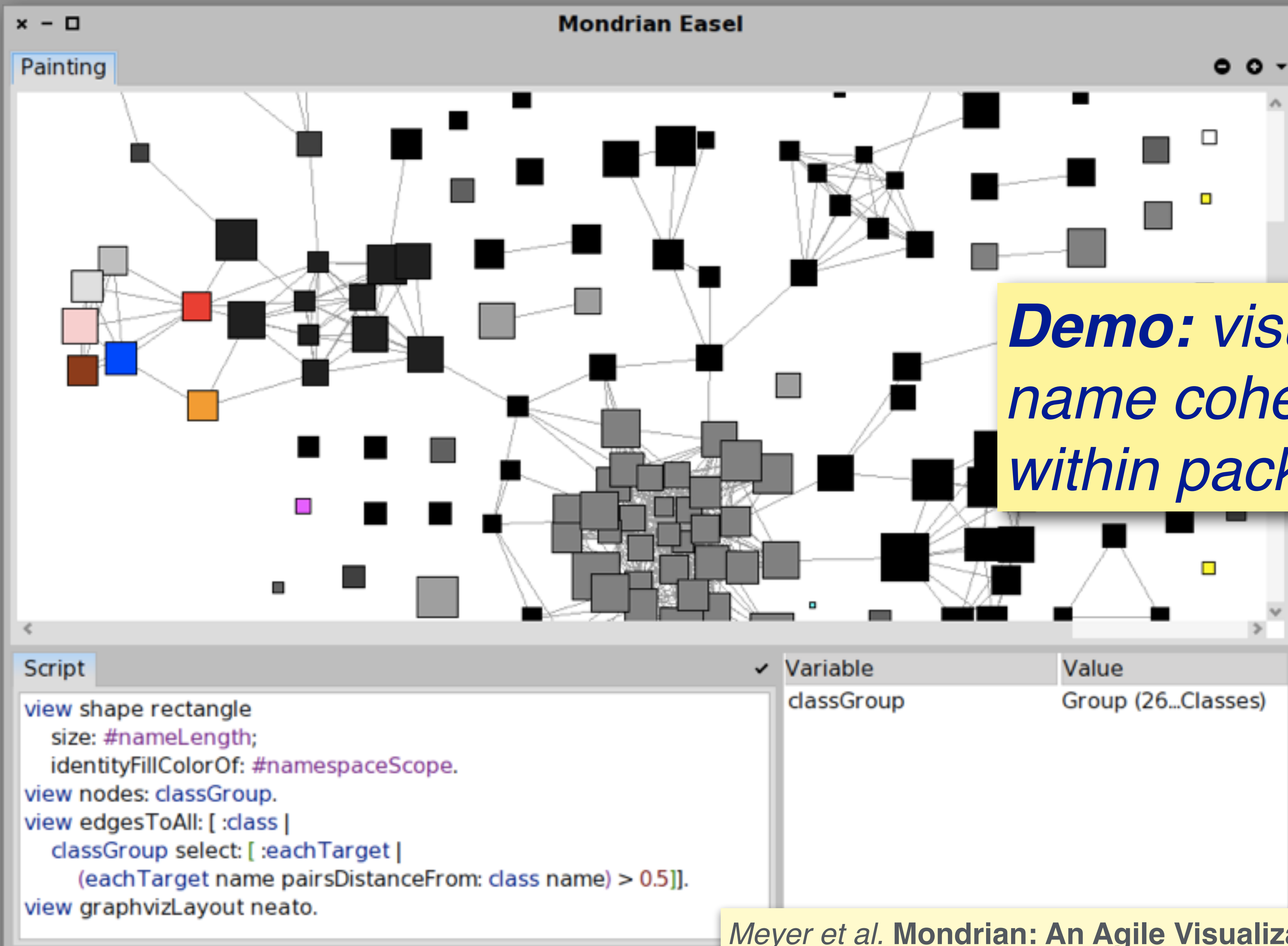


System complexity - Clone evolution view
Class blueprint - Topic Correlation Matrix - Distribution Map for topics spread over classes in packages
Hierarchy Evolution view - Ownership Map





Mondrian Demo



Demo: *visualizing name cohesion within packages*

Meyer et al. **Mondrian: An Agile Visualization Framework**. SoftVis 2006. DOI: 10.1145/1148493.1148513

Agile Modeling

Smalltalk

Java
Python
C++
...

Orion | DSM | BugMap | ...

Roassal

Extensible meta model

Model repository

Navigation
Metrics
Querying
Grouping

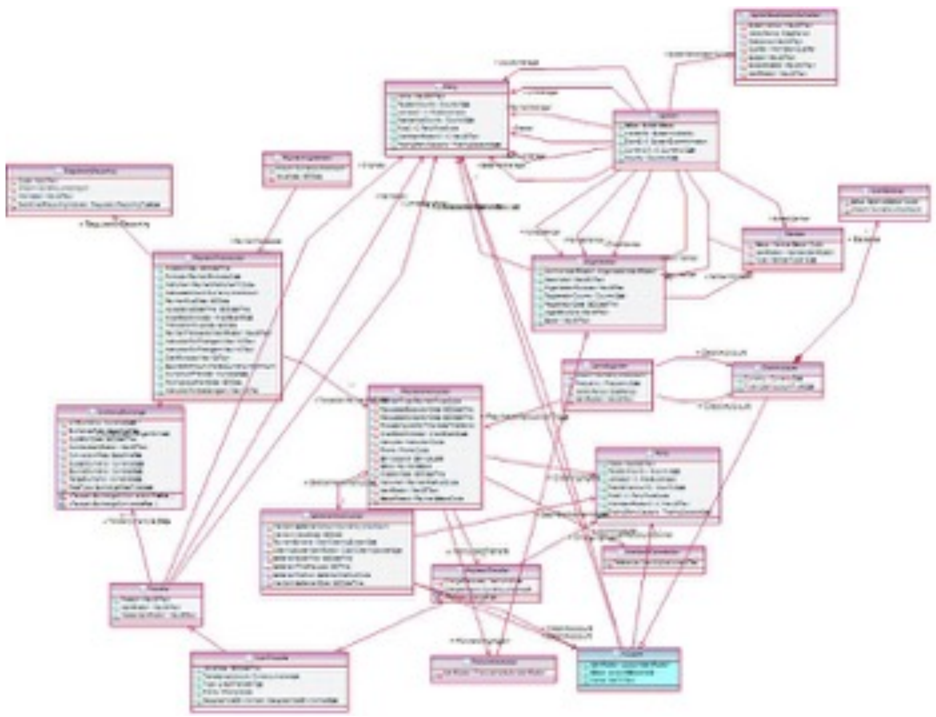
Smalltalk

Moose is a powerful tool once we have a model …

Nierstrasz et al. **The Story of Moose.**
ESEC/FSE 2005. DOI: 10.1145/1095430.1081707

Challenge

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



The key bottleneck to assessment is creating a suitable model for analysis. If a tool does not already exist, it can take days, weeks or months to parse source files and generate models.

Problems

Unknown languages

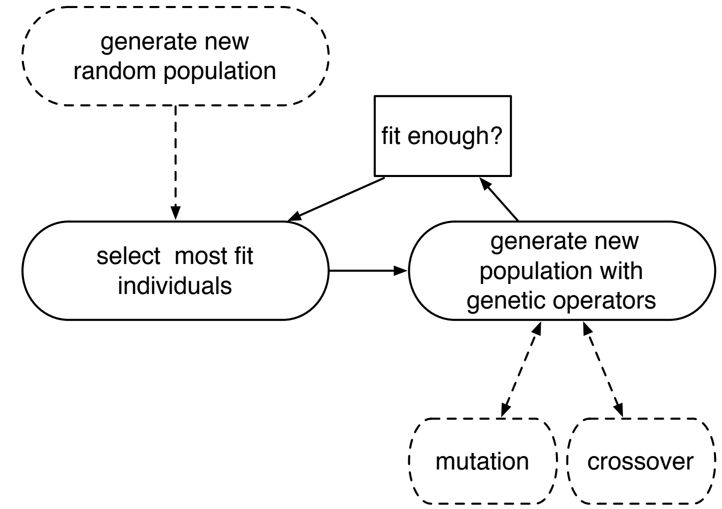
Heterogeneous projects

Unstructured text

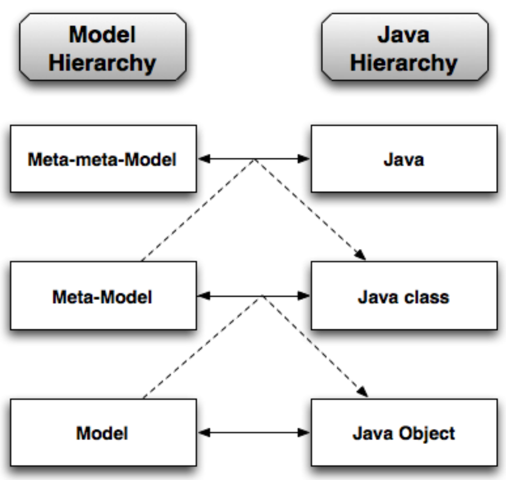
Developing a parser for a new language is a big challenge.
Parsers may be hard to scavenge from existing tools.
Not only source code, but other sources of information, like bug reports and emails can be invaluable for model building.
Few projects today are built using a single language. Often a GPL is mixed with scripting languages, or even home-brewed DSLs.

Ideas

Grammar Stealing

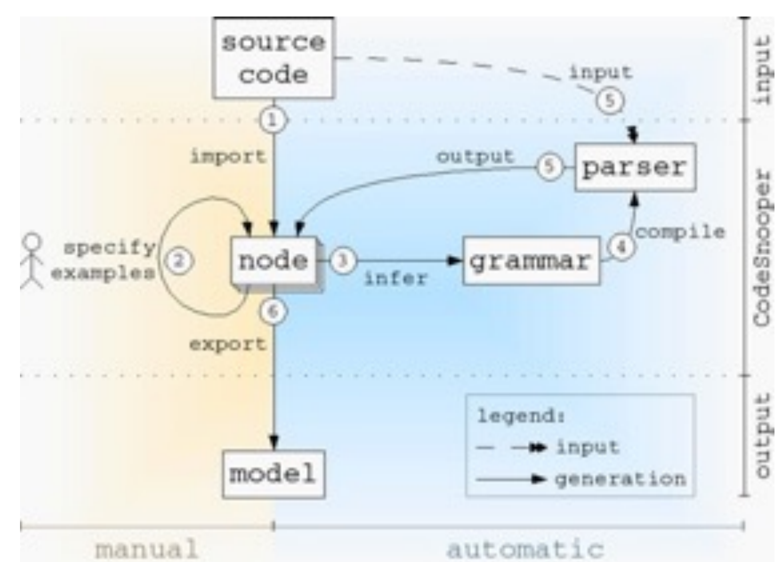


Evolutionary Grammar Generation



Recycling Trees

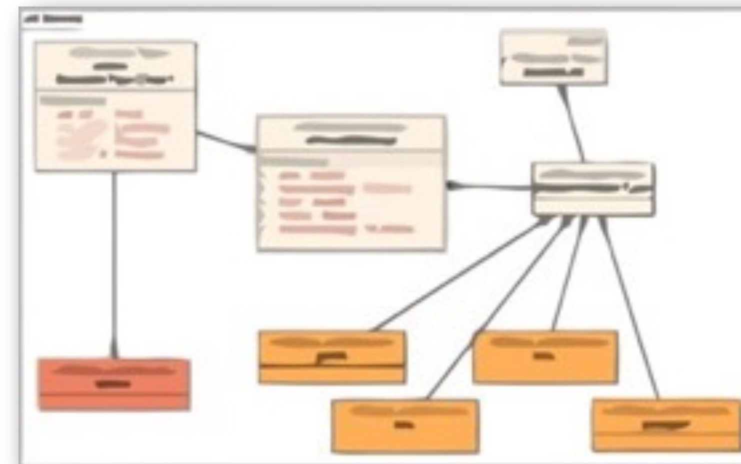
Hooking into an existing tool



Parsing by Example

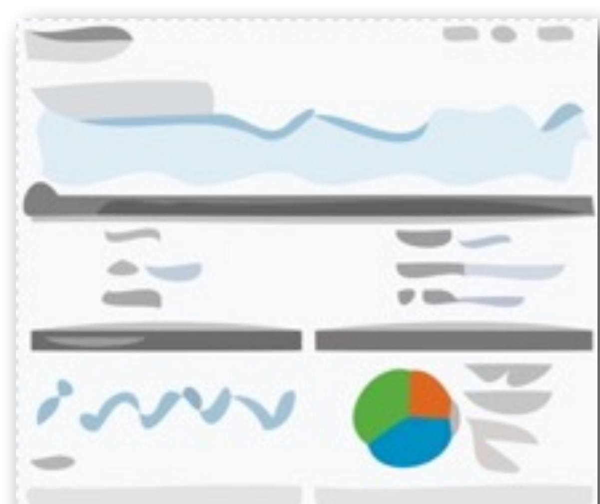
Agile Modeling Lifecycle

Build a
coarse model



Refine the
model

Build a custom
analysis



Idea: use island grammars to extract coarse models

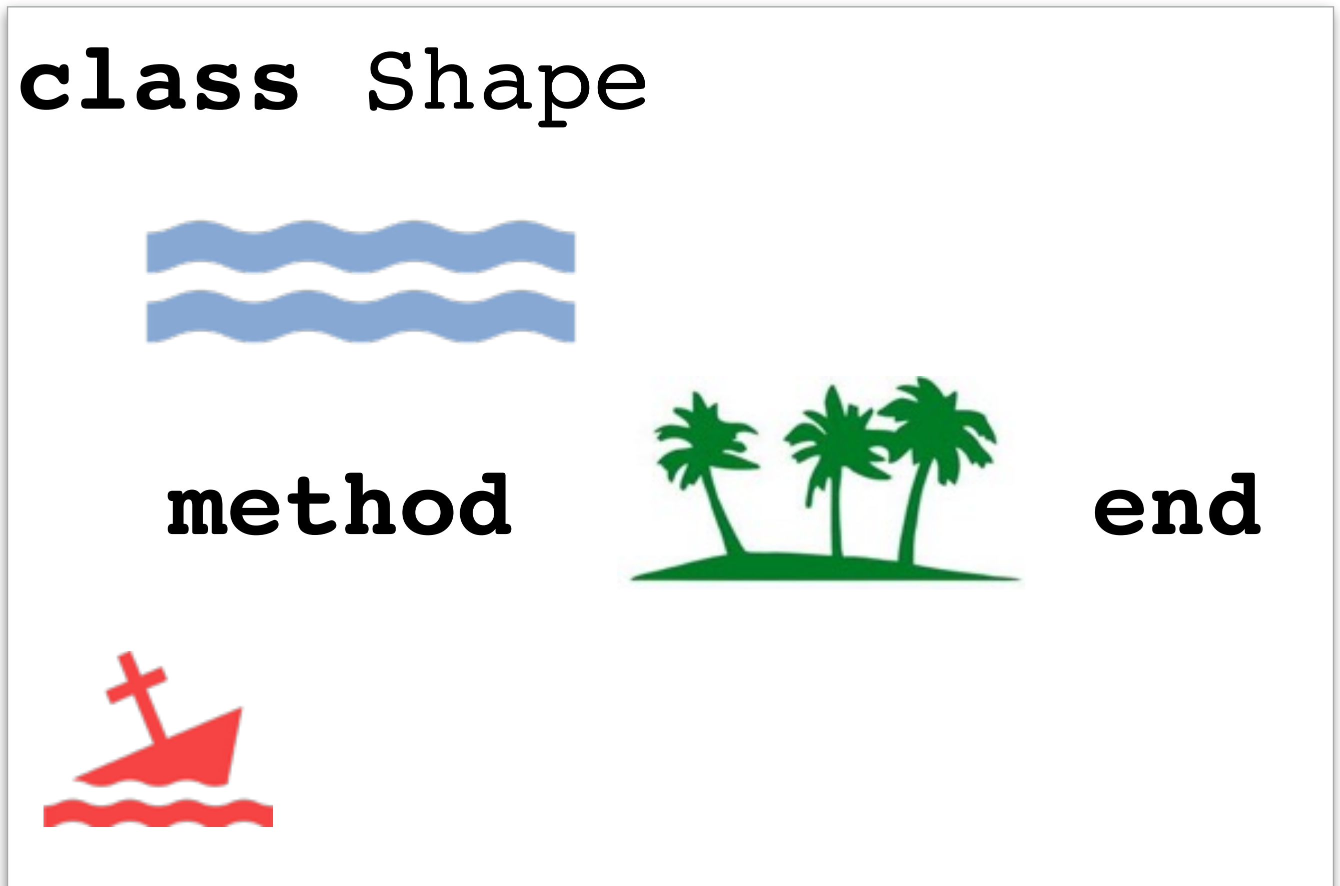
```
class Shape
   int x;
   int y;

   method draw() … end
end

method main() … end
```

```
'class' ID
    (method / . {avoid})*
'end'

method?
```

method → 

. {avoid} → 

Problem: island grammars lead to shipwrecks



Tweaking island grammars till they work is not an option …

```
'class' ID
    (method / !'end' !method)*
'end'

method?
```

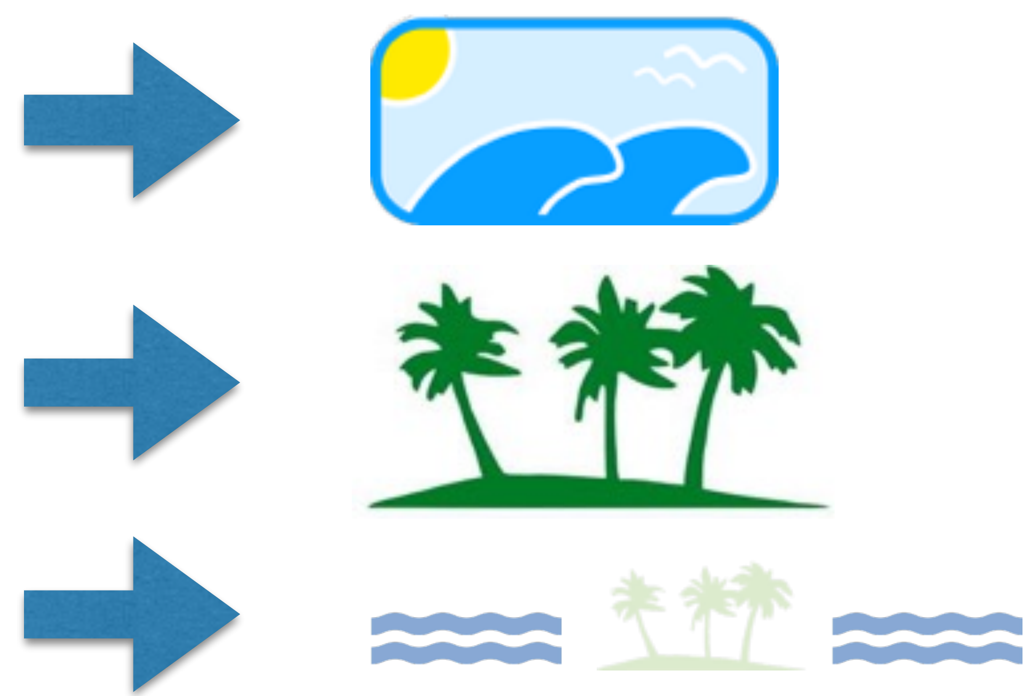
A Bounded Sea searches for an island in a bounded scope

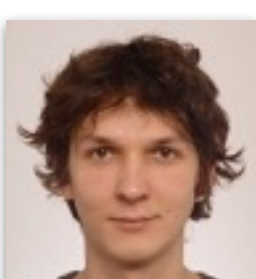
```
'class' ID
    (~method~)*
'end'

method?
```

~method~

method

~method~

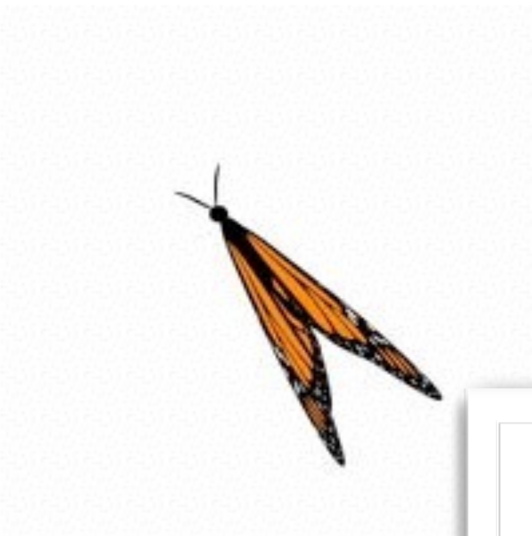




Jan Kurs, et al. **Bounded Seas: Island Parsing Without Shipwrecks.** SLE 2014. DOI: 10.1007/978-3-319-11245-9_4

Architectural Monitoring

Challenge



“What will my code change impact?”

Large software systems are so complex that one can never be sure until integration whether certain changes can have catastrophic effects at a distance.
Ideas: Tracking Software Architecture; exploiting Big Software Data

Problems

Diverse views of SA

SA is not in the code

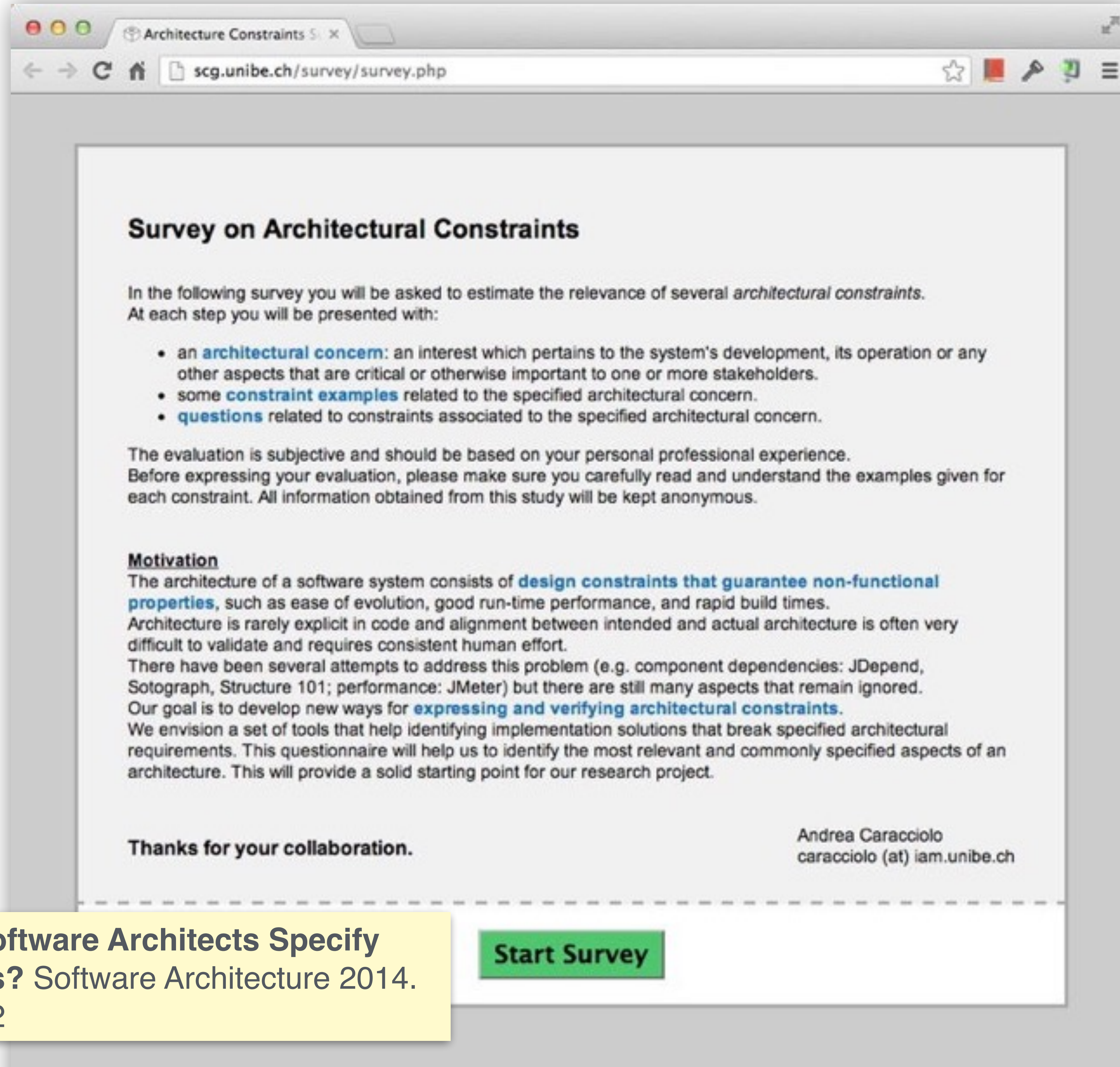
The IDE focuses on code

Ideas

Uncovering "Software Architecture in the Wild"

Architecture monitoring (beyond dependencies)

What is SA in the Wild?

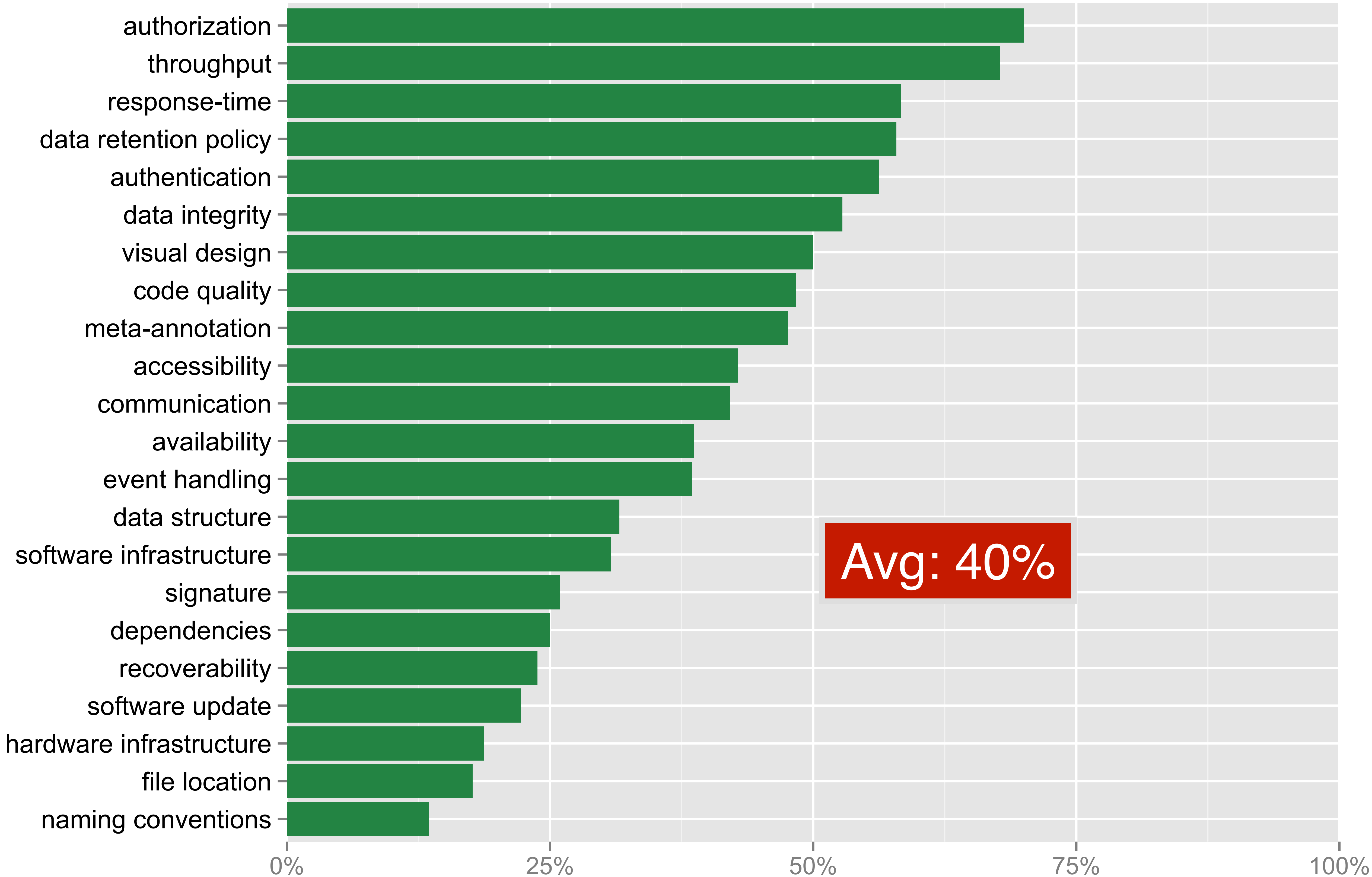


Andrea Caracciolo, et al. **How Do Software Architects Specify and Validate Quality Requirements?** Software Architecture 2014. DOI: 10.1007/978-3-319-09970-5_32

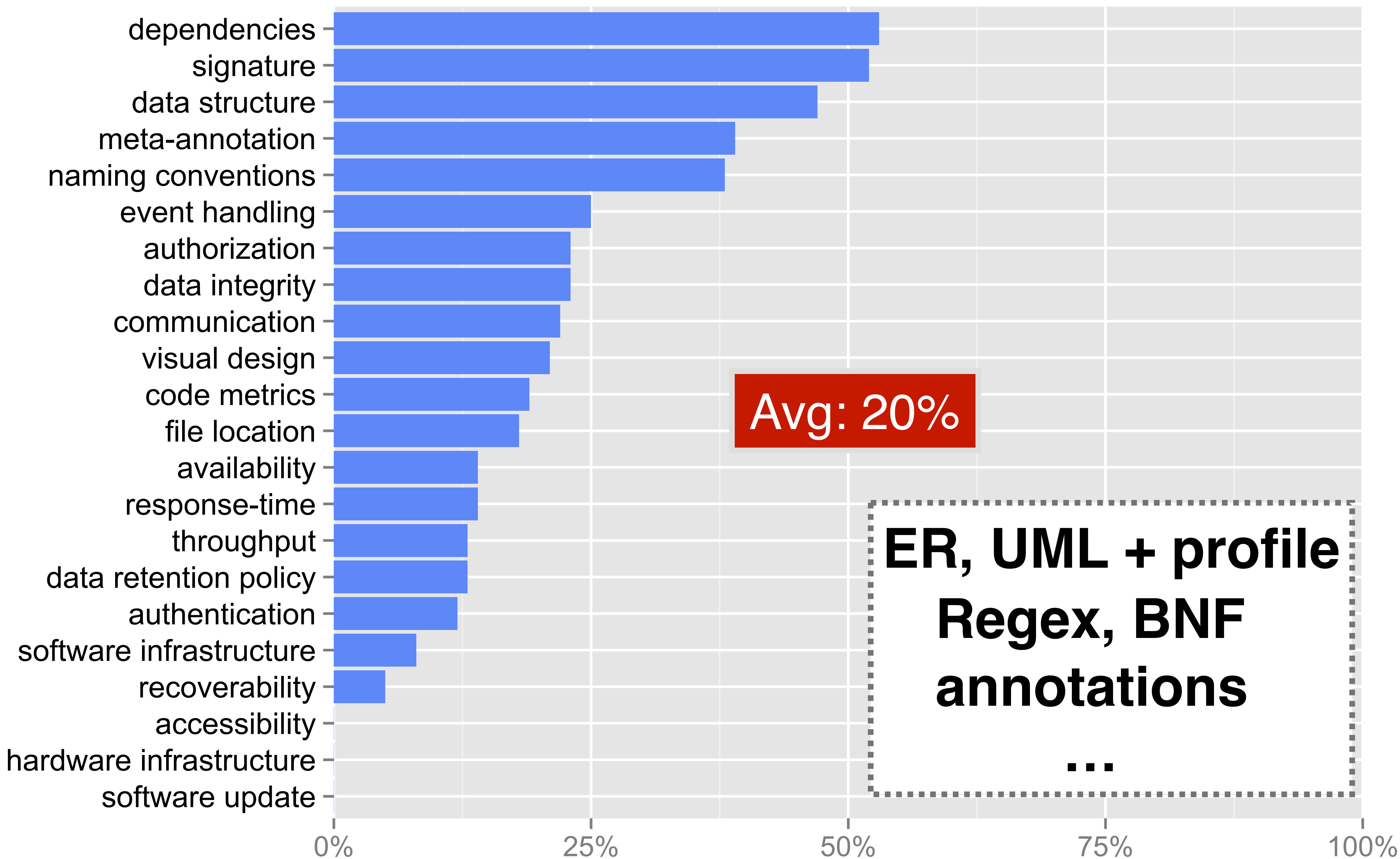
Impact of SA constraints

constraint	Impact (1-5)
availability	4.2
response-time	4.0
authorization	3.9
authentication	3.6
communication	3.4
throughput	3.4
signature	3.4
software infrastructure	3.3
data integrity	3.3
recoverability	3.1
dependencies	3.1
visual design	3.0
data retention policy	3.0
hardware infrastructure	2.9
system behavior	2.9
data structure	2.9
event handling	2.9
code metrics	2.7
meta-annotation	2.6
naming conventions	2.6
file location	2.5
accessibility	2.5
software update	2.2

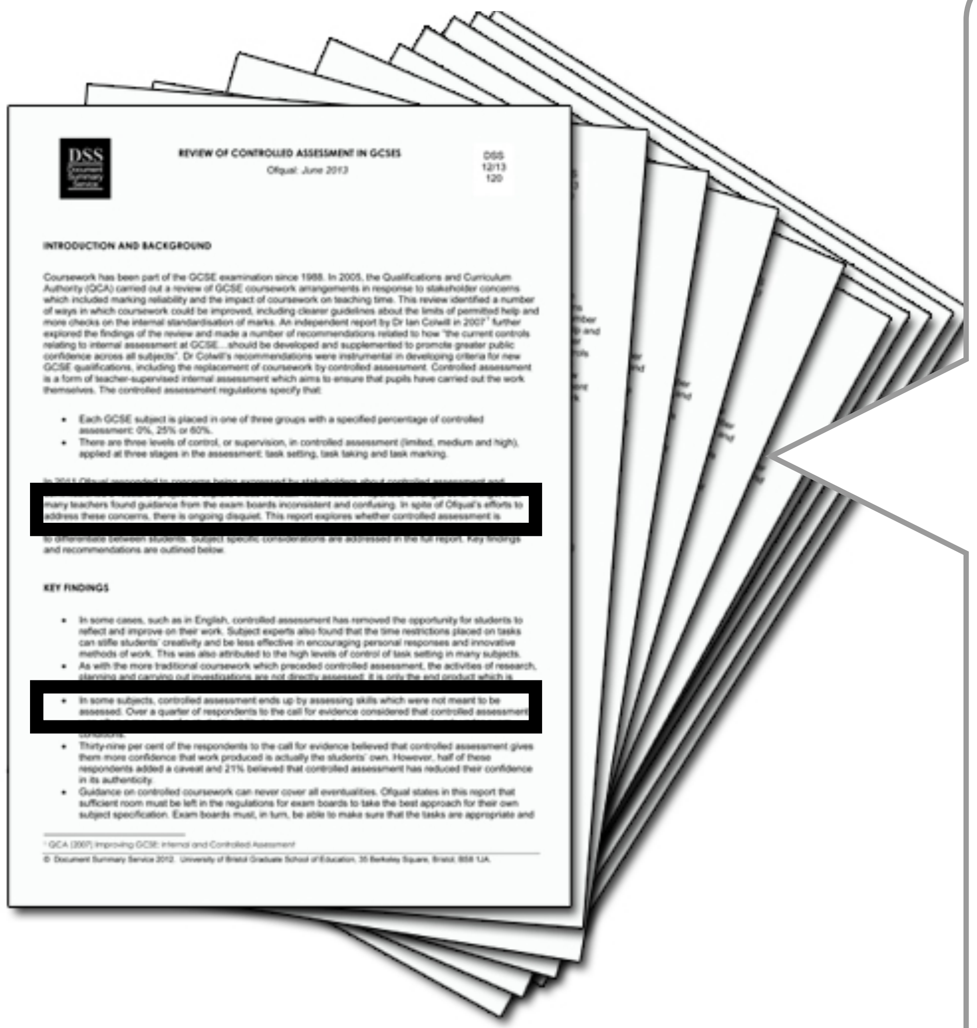
Automated Validation is not Prevalent



Formalization is not Prevalent



Architectural Rules



Naming Conventions

“Repository interfaces can only declare methods named find..()”

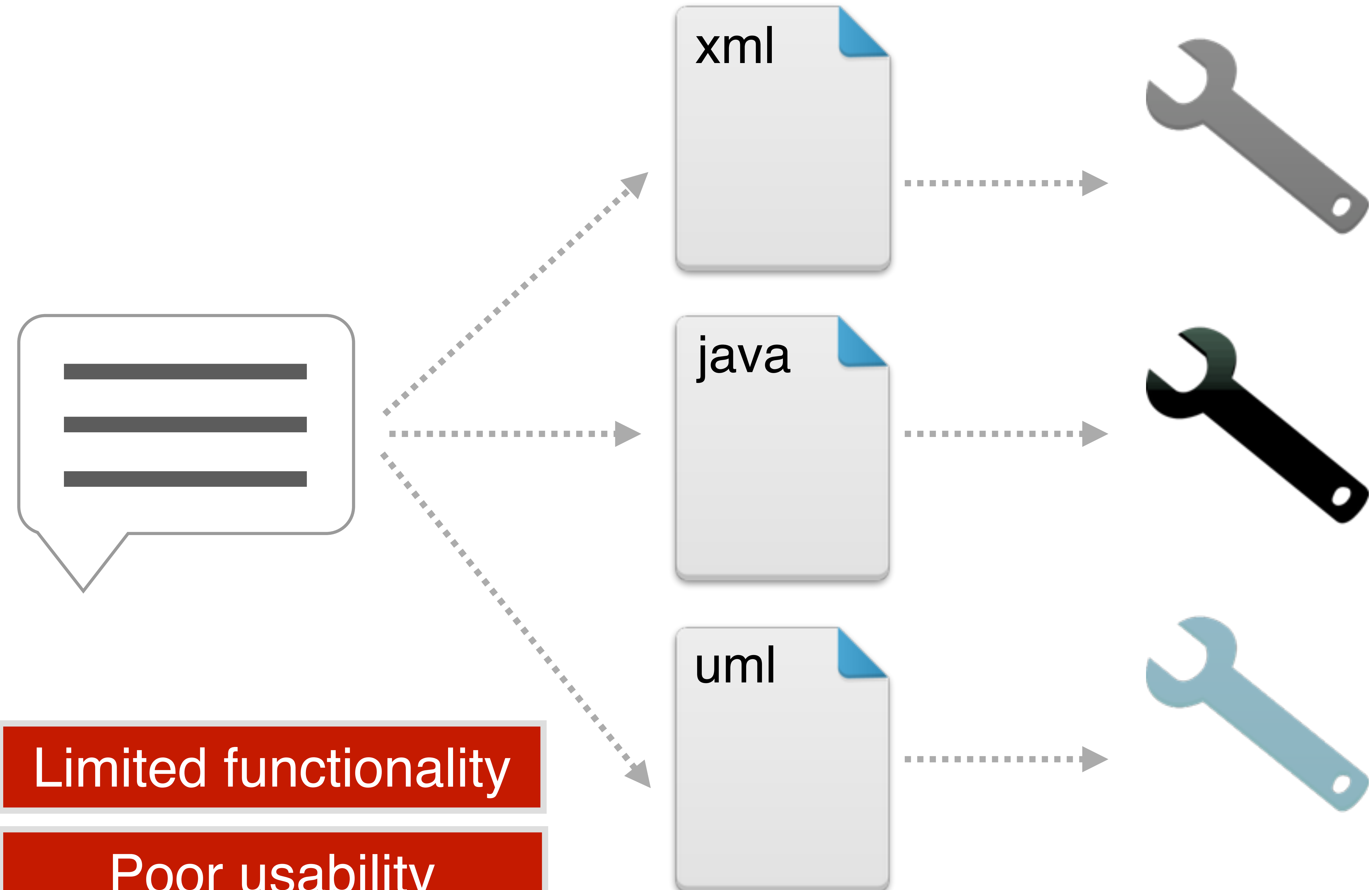
Dependencies

“Only Service classes are allowed to throw AppException”

Performance

“The rendering operation has to be completed in less than 4ms”

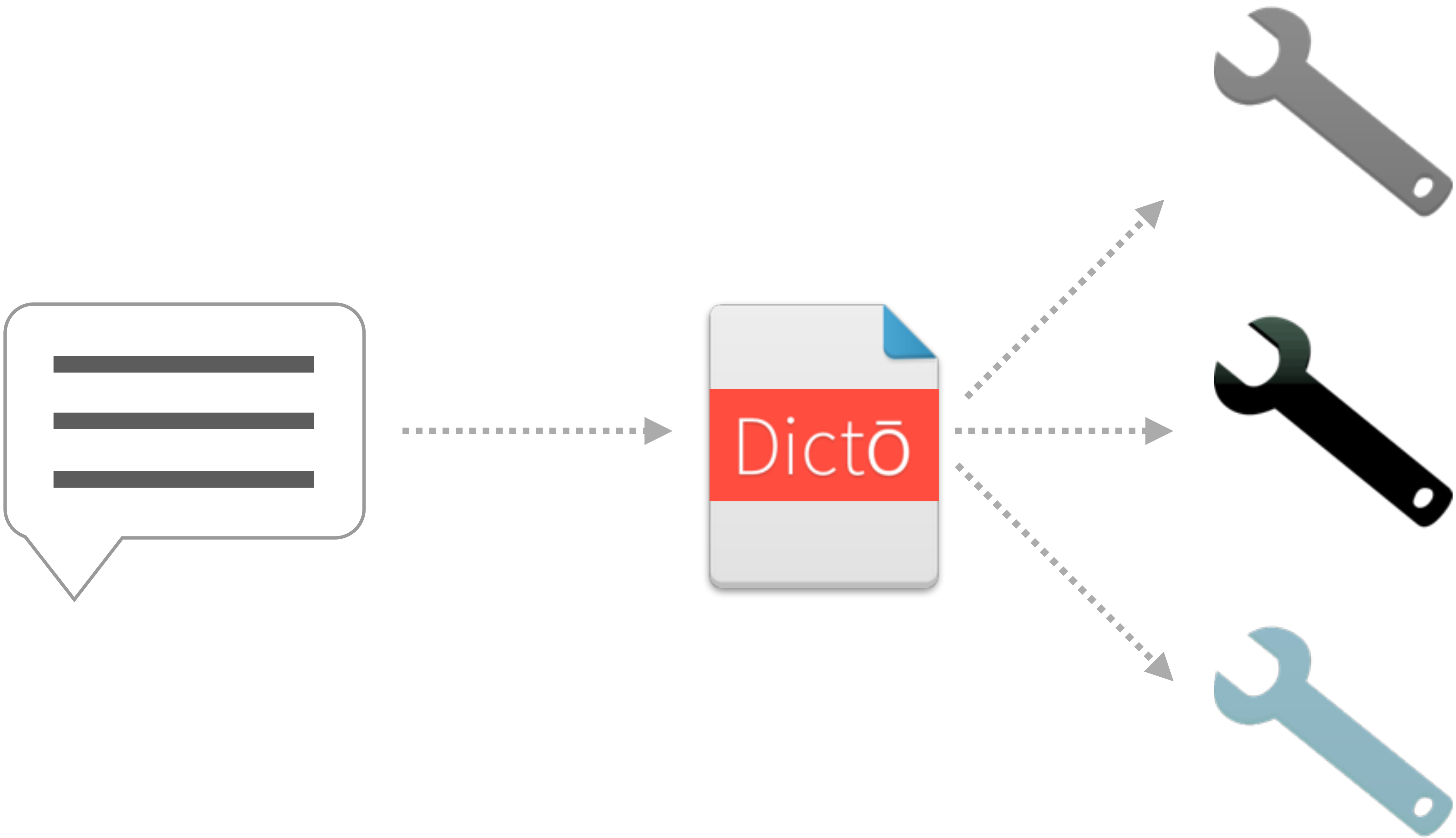
Rule Validation



Limited functionality

Poor usability

Dicto — a unified ADSL



Andrea Caracciolo, et al. **Dicto: A Unified DSL for Testing Architectural Rules.** ECSAW '14. DOI: 10.1145/2642803.2642824

Dicto Rules

…

`MyService : Website with url=“http://www.abc.com/api”`

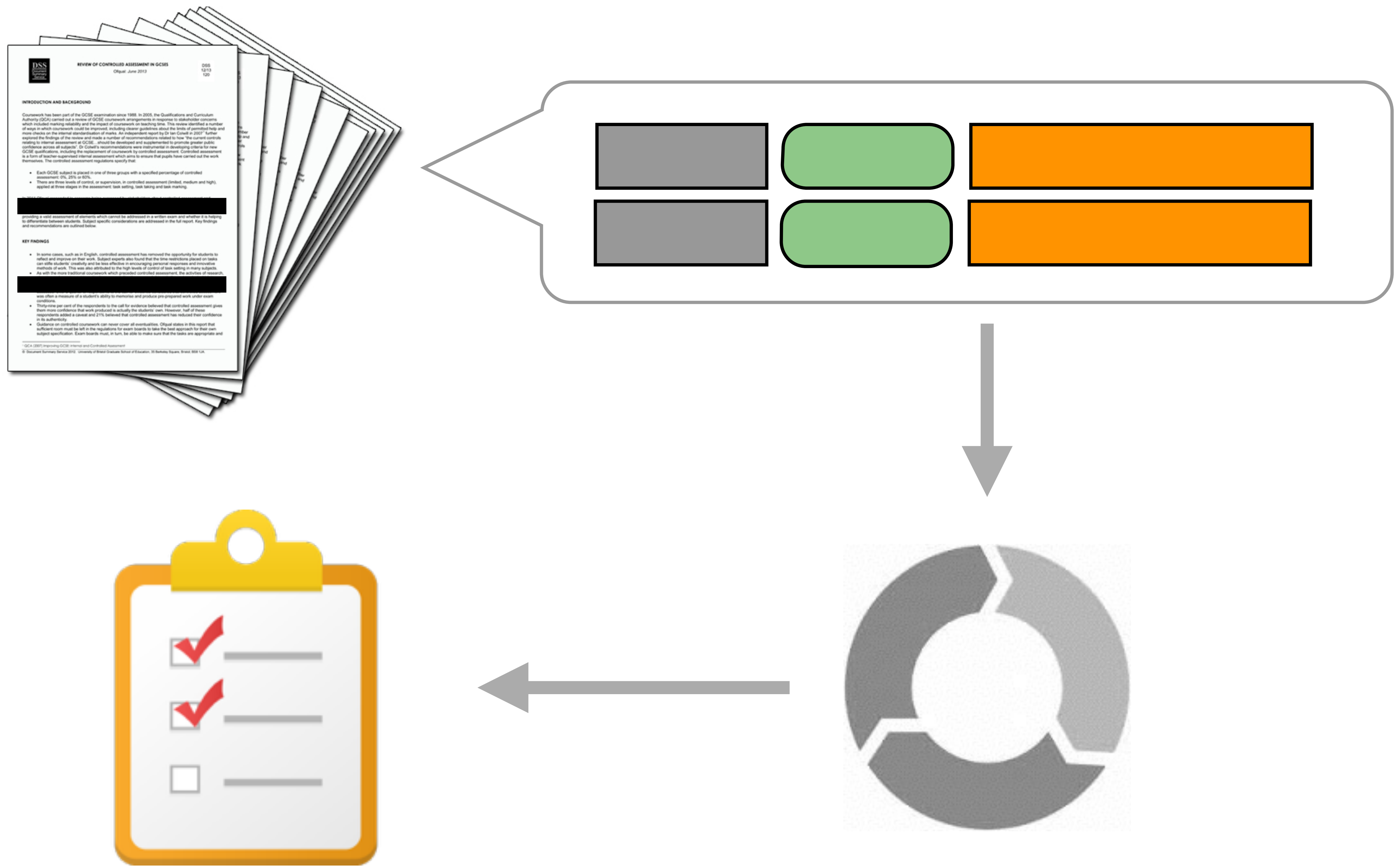
`MyService` **`must`** `HandleLoadFrom("10 users")`

`MyService` **`cannot`** `HaveResponseTimeLessThan(“1000 ms")`

`MyService` **`can only`** `HandleSOAPMessages()`

…

Periodic Validation



Rule Examples

Website response time
Website load testing

Dependencies

Code clones

Deadlock freeness

File Content

grep

Moldable Tools

Challenge

Build a new assessment tool in ten minutes

Custom analyses require custom tools. Building a tool should be as easy as writing a query in SQL or a form-based interface.

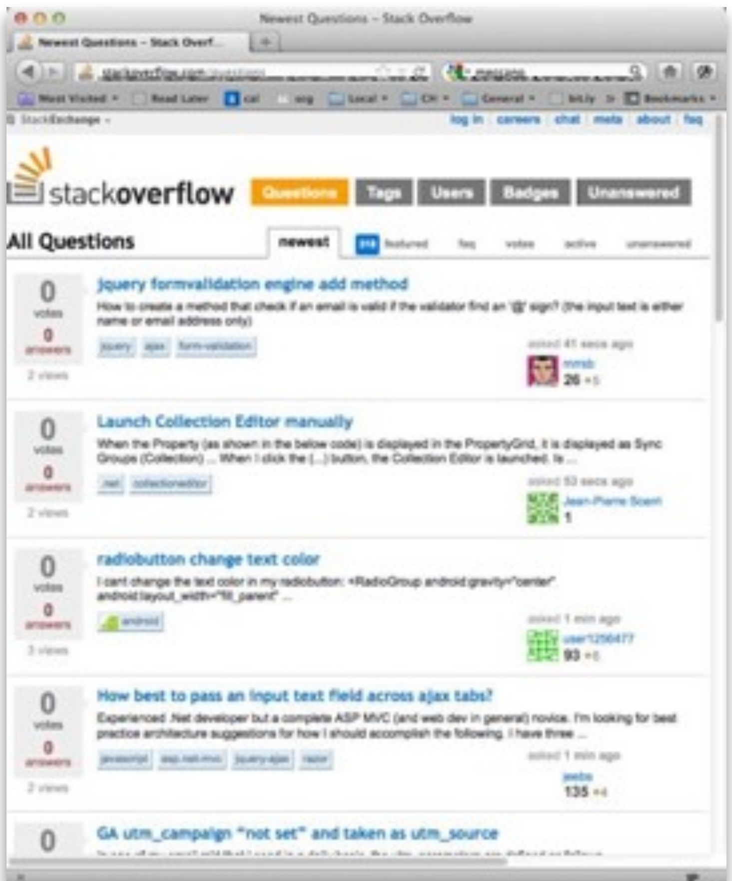
Problems

What tools do developers really need?

What is a unifying meta-model for tool construction?

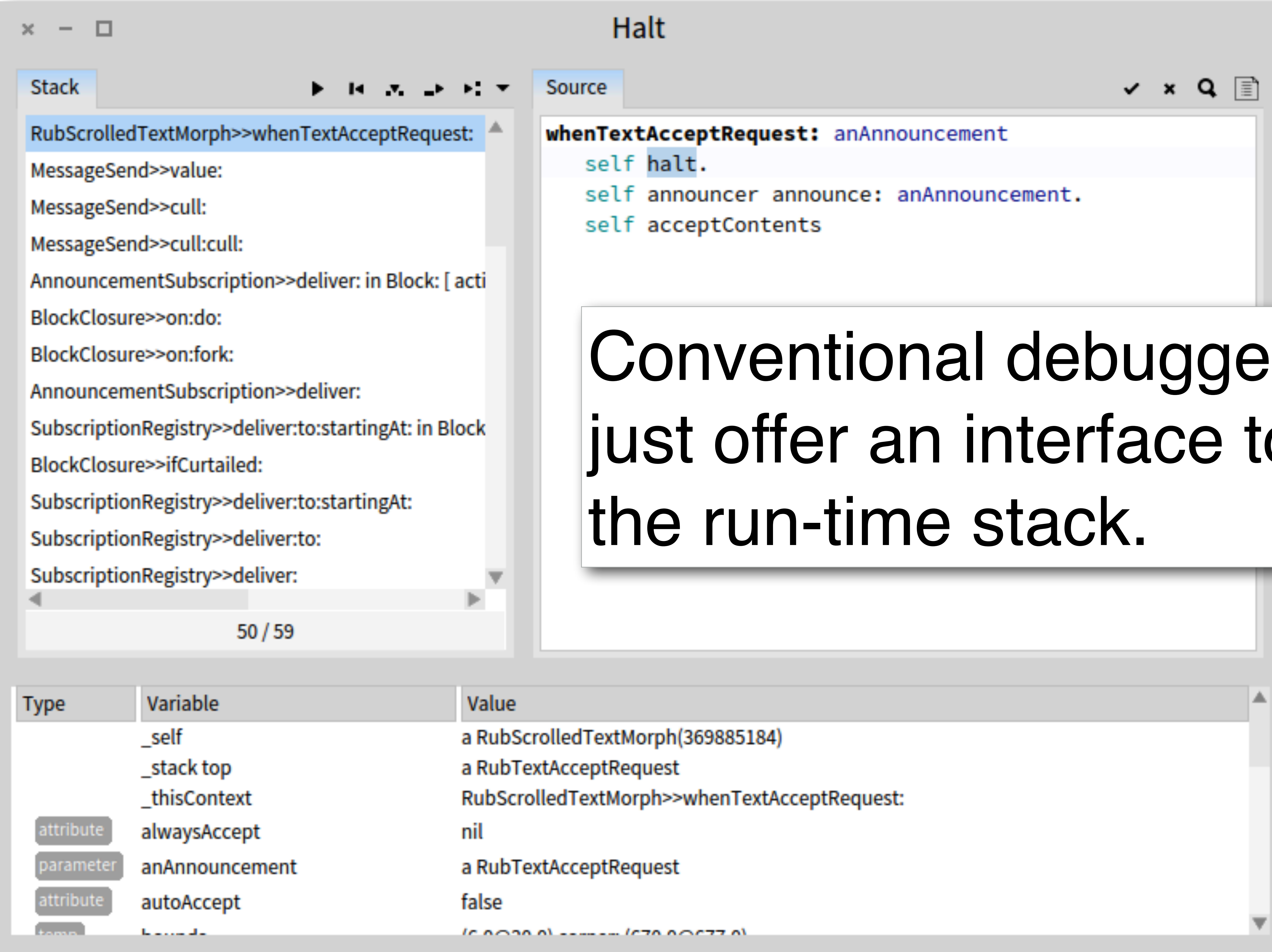
What are appropriate meta-tools?

Ideas



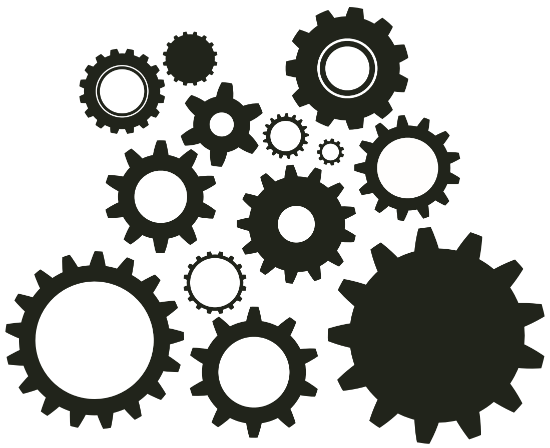
Analyze developer needs (!)

"Moldable" Tools (not just plug-ins)

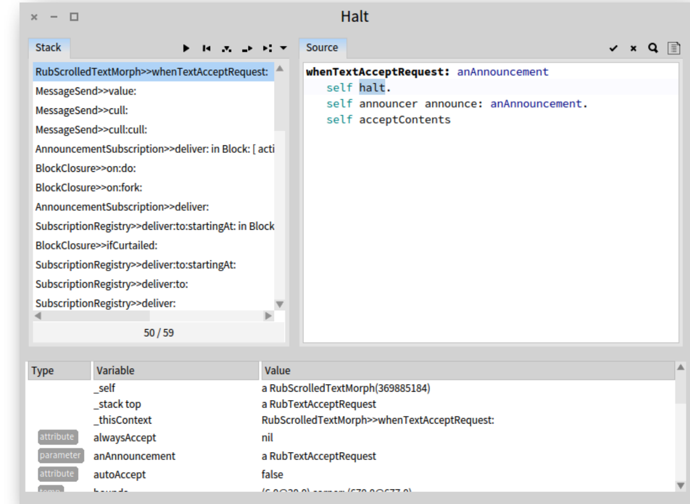


Conventional debuggers just offer an interface to the run-time stack.

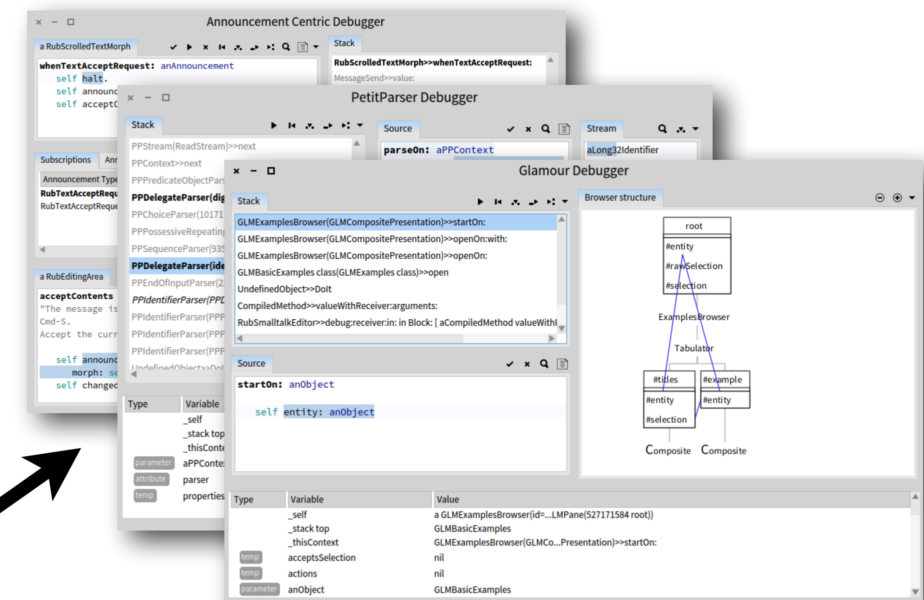
Specific Models



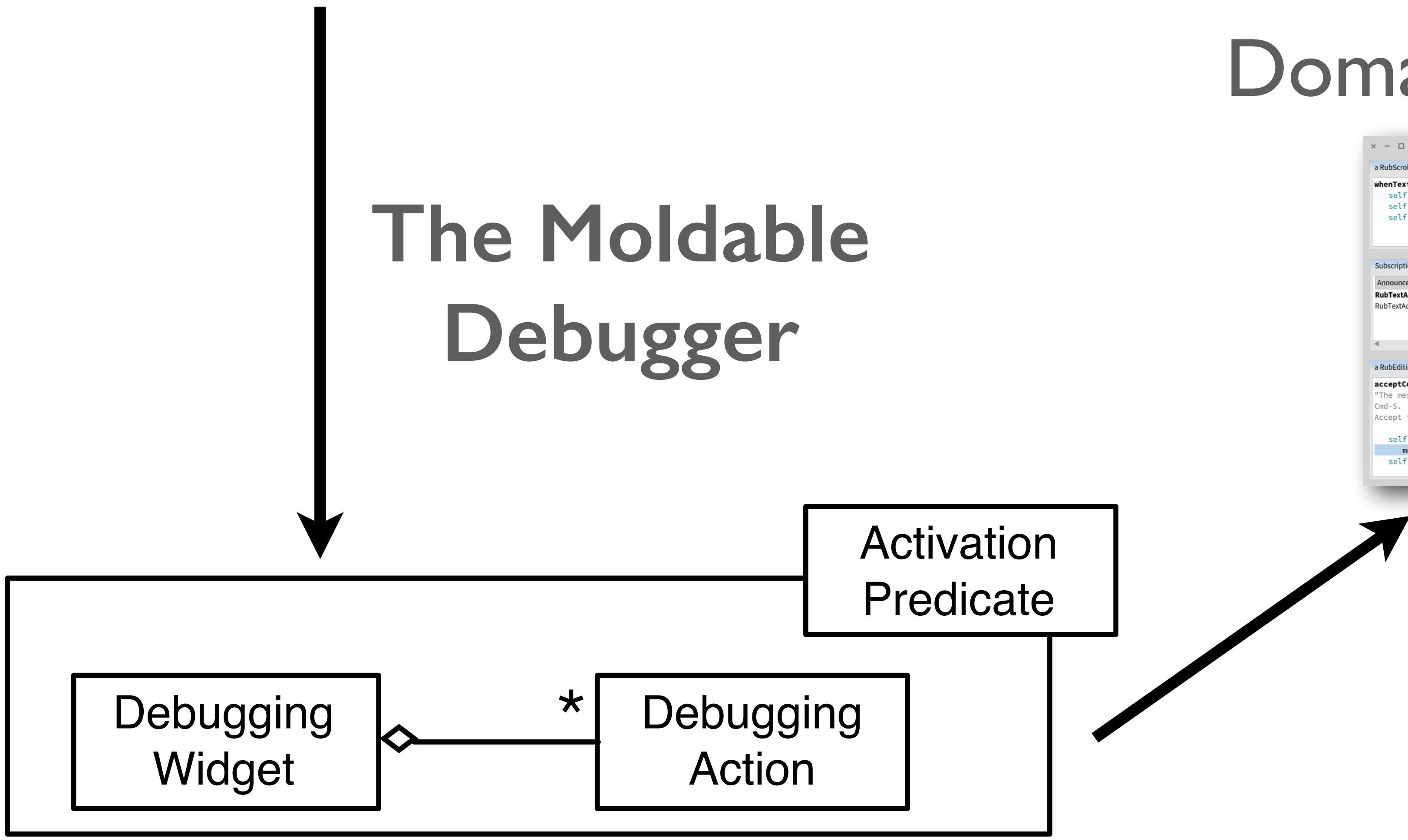
Generic Debugger



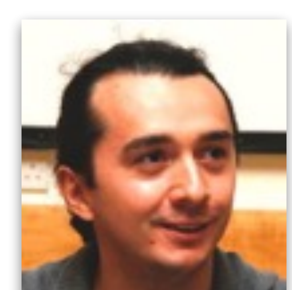
Domain-specific Debuggers



The Moldable Debugger

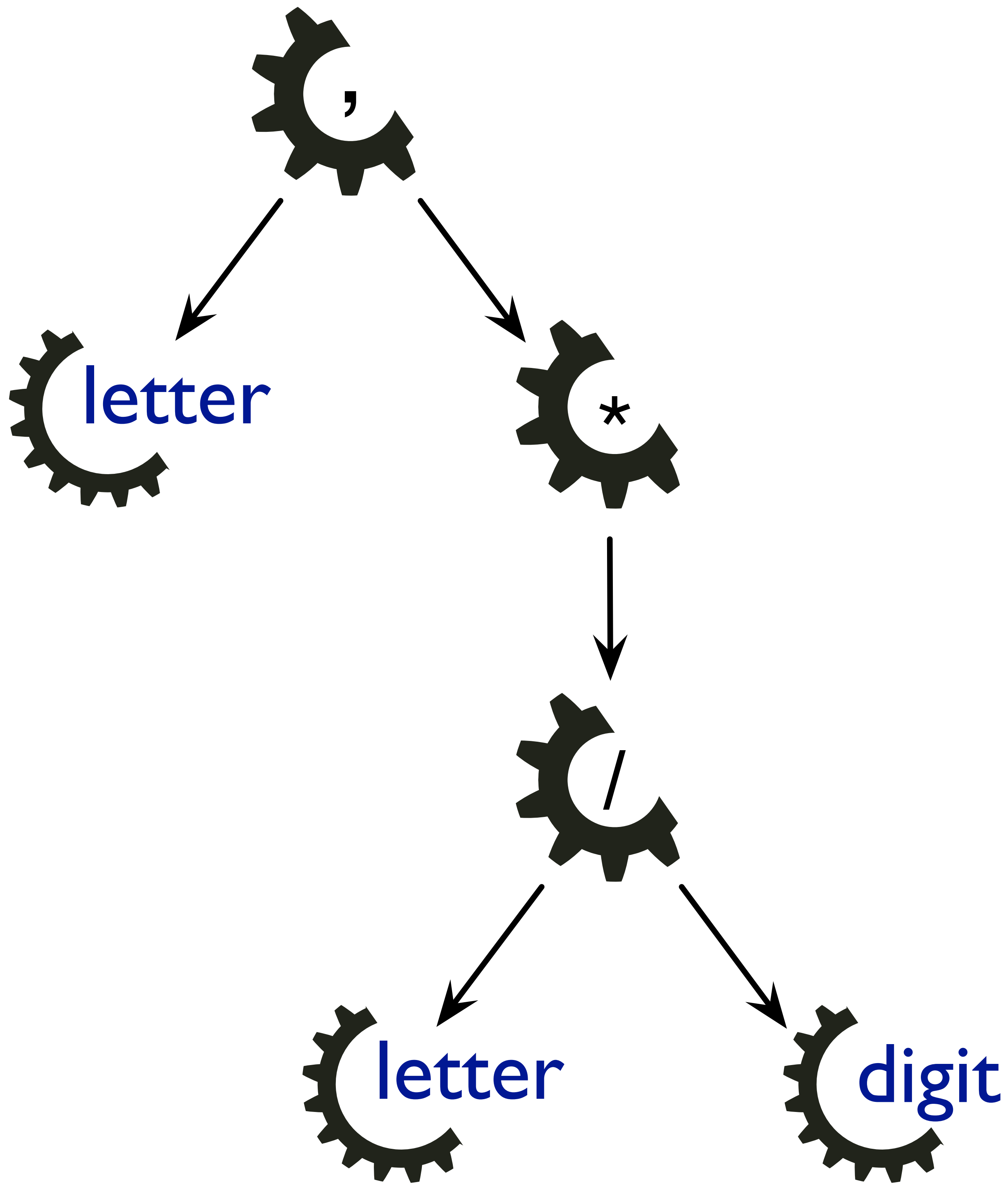


Andrei Chis et al. **The Moldable Debugger: A Framework for Developing Domain-Specific Debuggers.** SLE 2014. DOI: 10.1007/978-3-319-11245-9_6

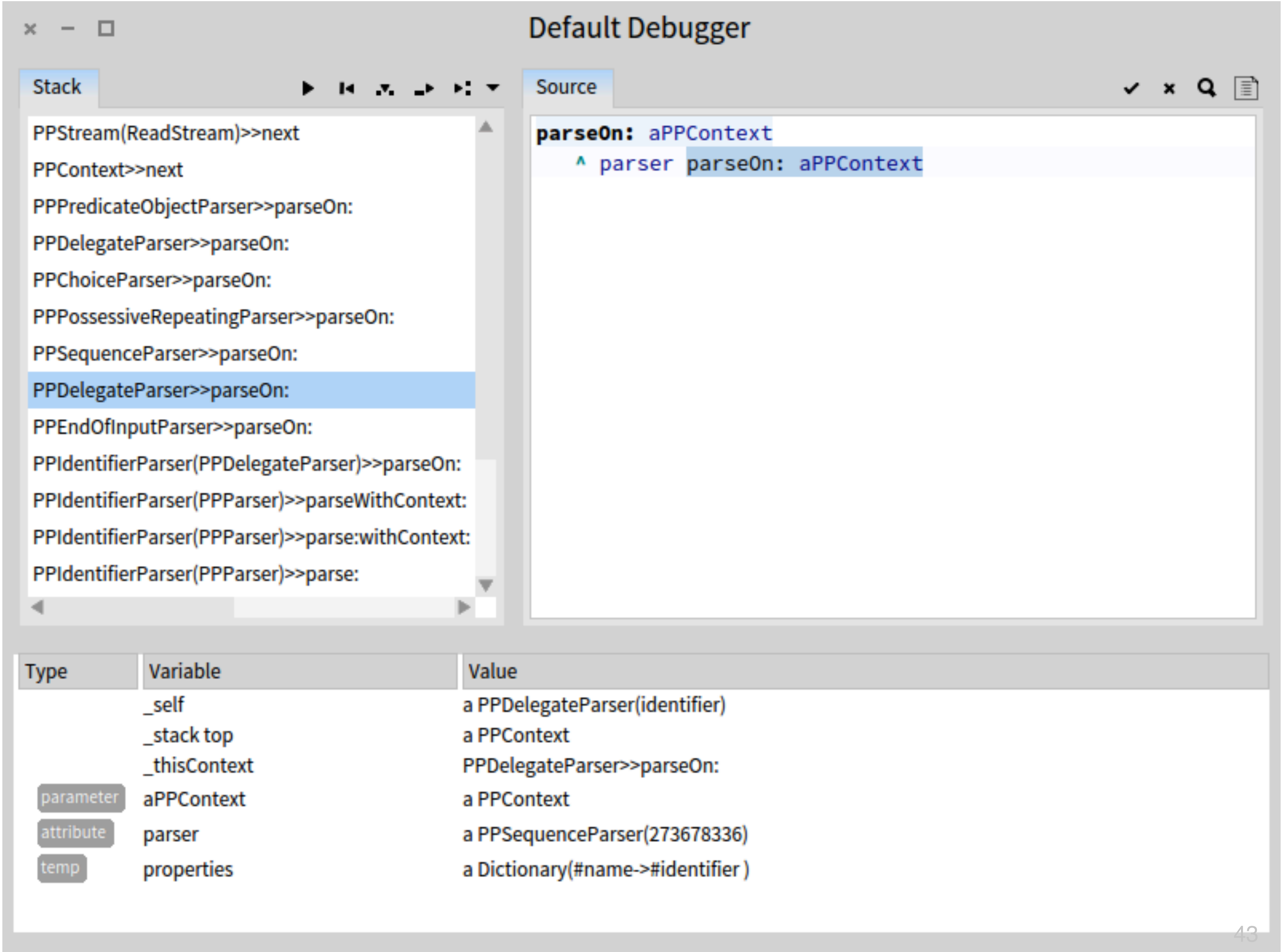


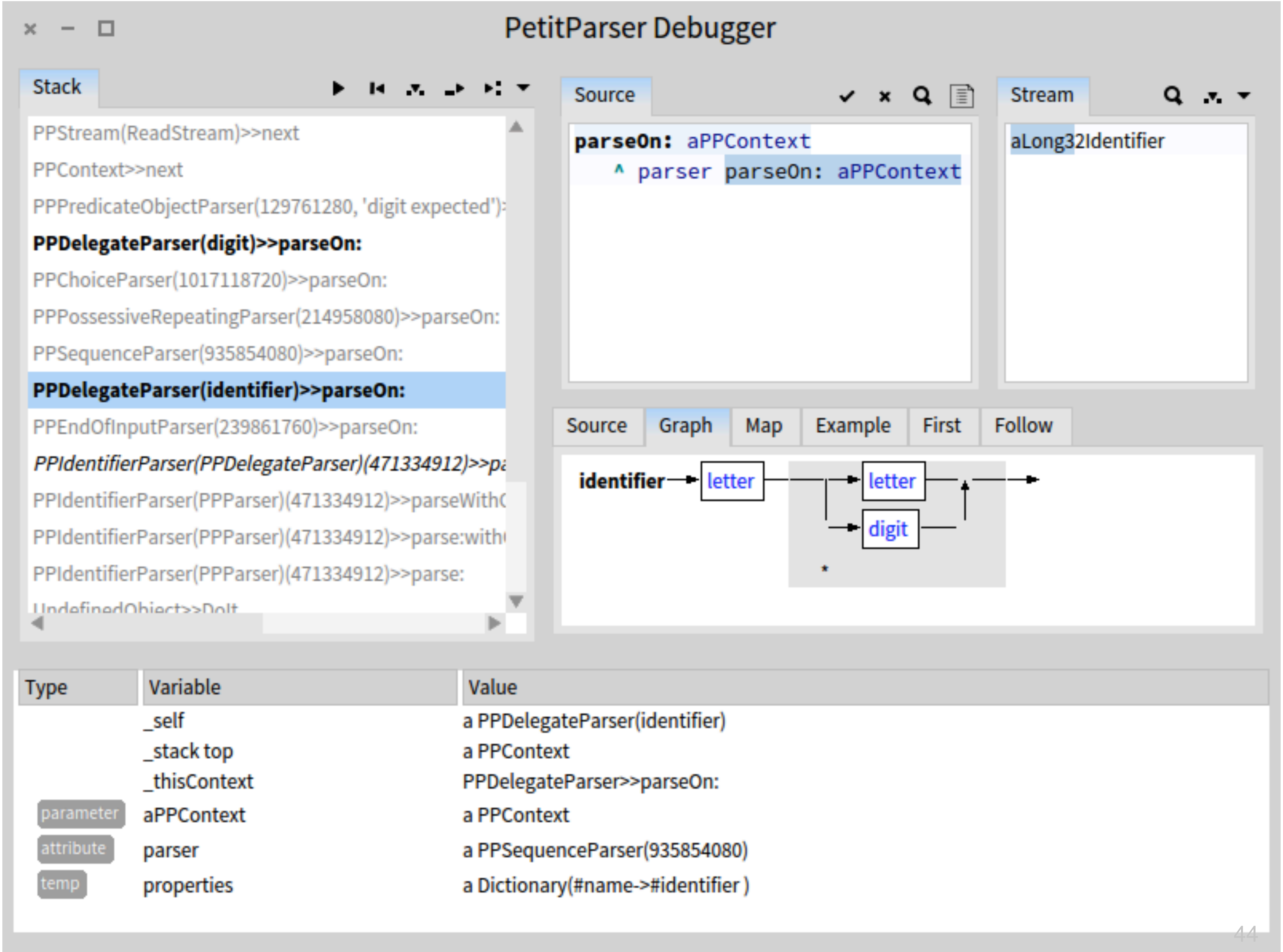
PetitParser

identifier
    letter , (letter / digit) *

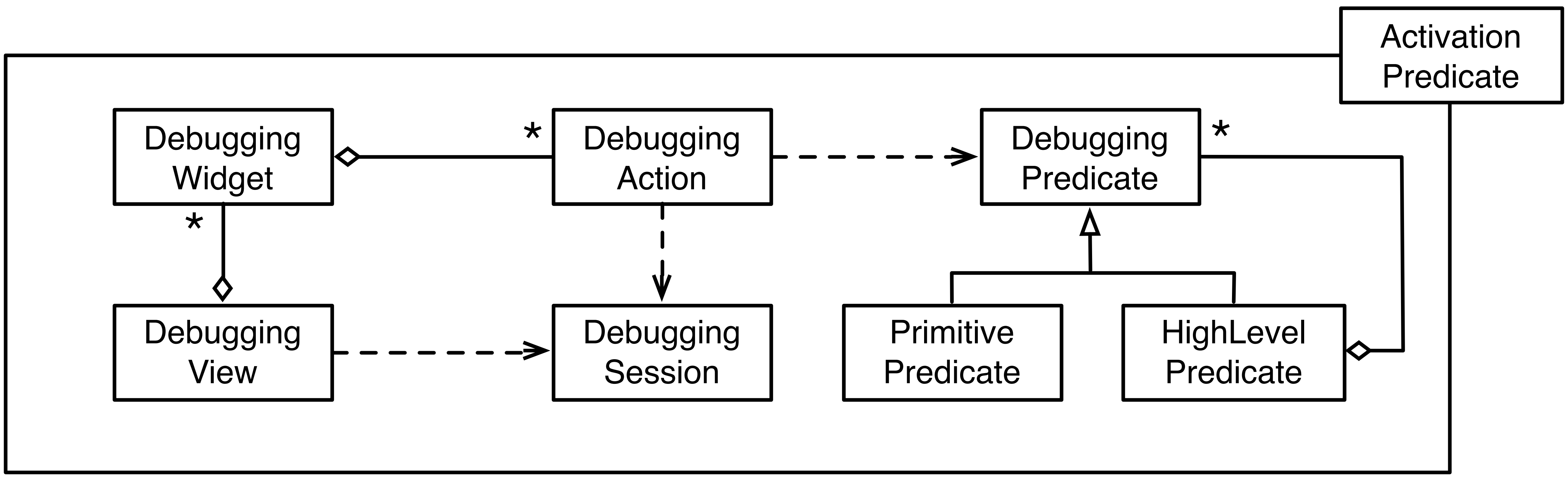


```
IdentifierParser new
    parse: 'aLong32Identifier'
```

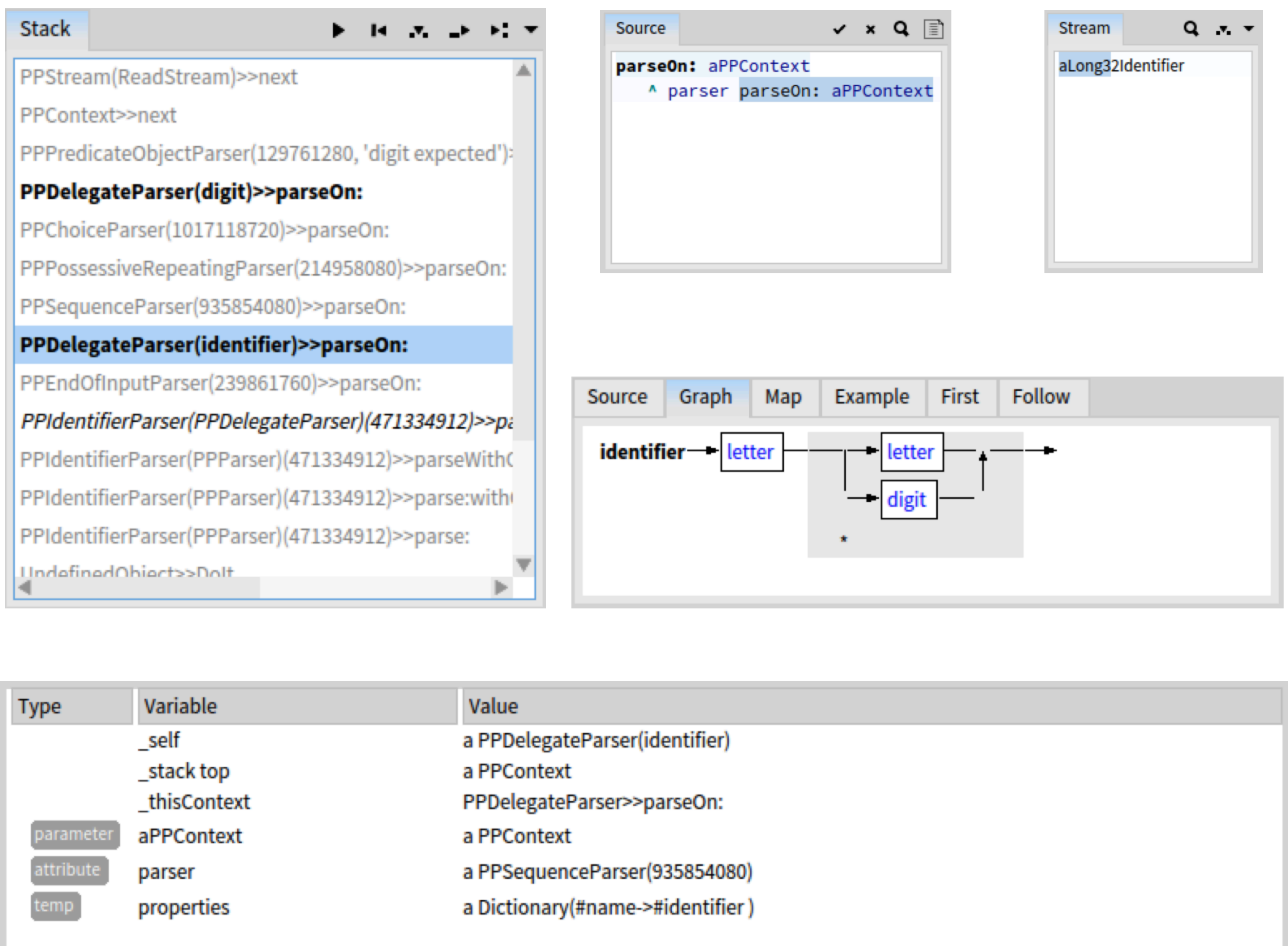

Default Debugger
Stack
PPStream(ReadStream)>>next
PPContext>>next
PPPredicateObjectParser>>parseOn:
PPDelegateParser>>parseOn:
PPChoiceParser>>parseOn:
PPPossessiveRepeatingParser>>parseOn:
PPSequenceParser>>parseOn:
PPDelegateParser>>parseOn:
PPEndOfInputParser>>parseOn:
PPIdentifierParser(PPDelegateParser)>>parseOn:
PPIdentifierParser(PPParser)>>parseWithContext:
PPIdentifierParser(PPParser)>>parse:withContext:
PPIdentifierParser(PPParser)>>parse:
Source
parseOn: aPPContext
^ parser parseOn: aPPContext
Type
Variable
Value
_self
a PPDelegateParser(identifier)
_stack top
a PPContext
_thisContext
PPDelegateParser>>parseOn:
parameter
aPPContext
a PPContext
attribute
parser
a PPSequenceParser(273678336)
temp
properties
a Dictionary(#name->#identifier)


PetitParser Debugger
Stack
PPStream(ReadStream)>>next
PPContext>>next
PPPredicateObjectParser(129761280, 'digit expected')
PPDelegateParser(digit)>>parseOn:
PPChoiceParser(1017118720)>>parseOn:
PPPossessiveRepeatingParser(214958080)>>parseOn:
PPSequenceParser(935854080)>>parseOn:
PPDelegateParser(identifier)>>parseOn:
PPEndOfInputParser(239861760)>>parseOn:
PPIdentifierParser(PPDelegateParser)(471334912)>>pa
PPIdentifierParser(PPParser)(471334912)>>parseWith
PPIdentifierParser(PPParser)(471334912)>>parse:with
PPIdentifierParser(PPParser)(471334912)>>parse:
Source
parseOn: aPPContext
^ parser parseOn: aPPContext
Stream
aLong32Identifier
Source
Graph
Map
Example
First
Follow
identifier
letter
letter
digit
*
Type
Variable
Value
_self
a PPDelegateParser(identifier)
_stack top
a PPContext
_thisContext
PPDelegateParser>>parseOn:
parameter
aPPContext
a PPContext
attribute
parser
a PPSequenceParser(935854080)
temp
properties
a Dictionary(#name->#identifier)

Domain specific-extensions



Debugging widgets



Debugging actions

Next parser

Next production

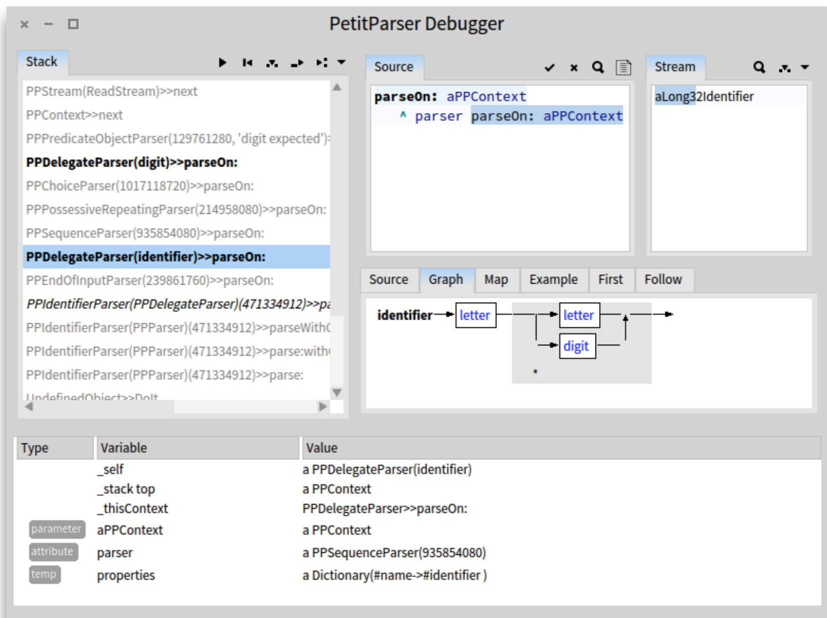
Production(aproduction)

Next failure

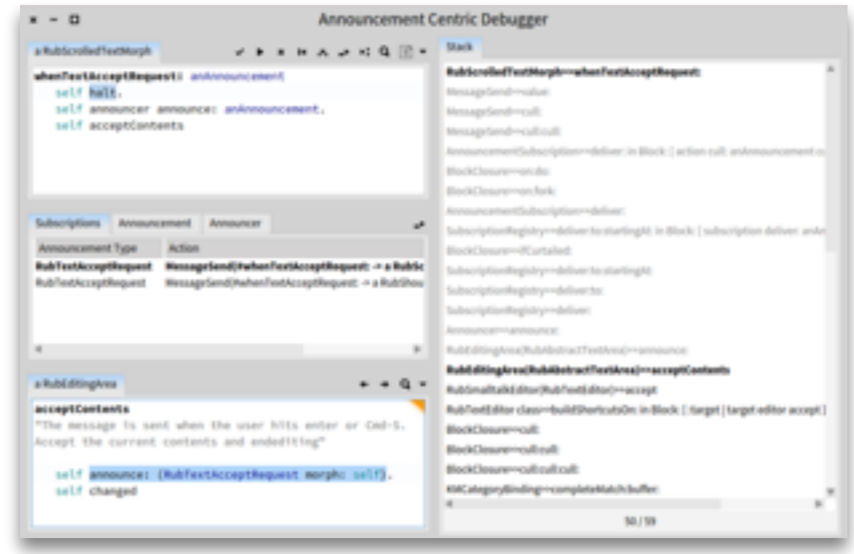
Stream position(anInteger)

Stream position changed

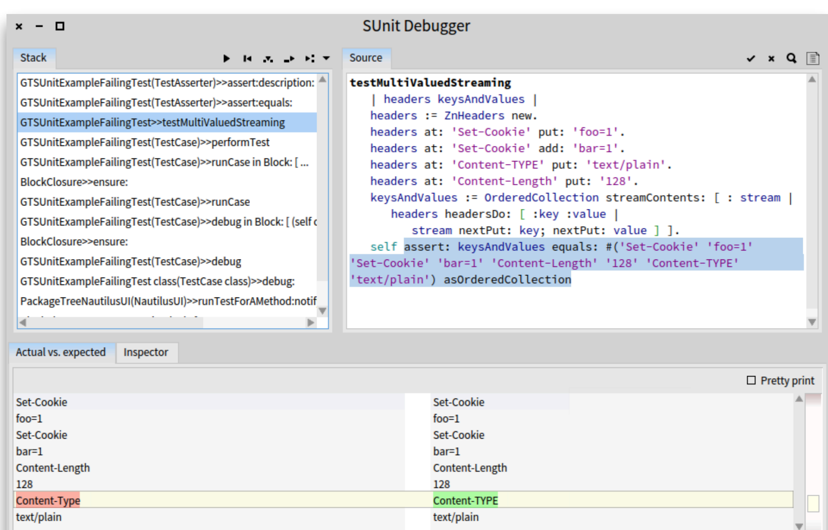
Petit Parser



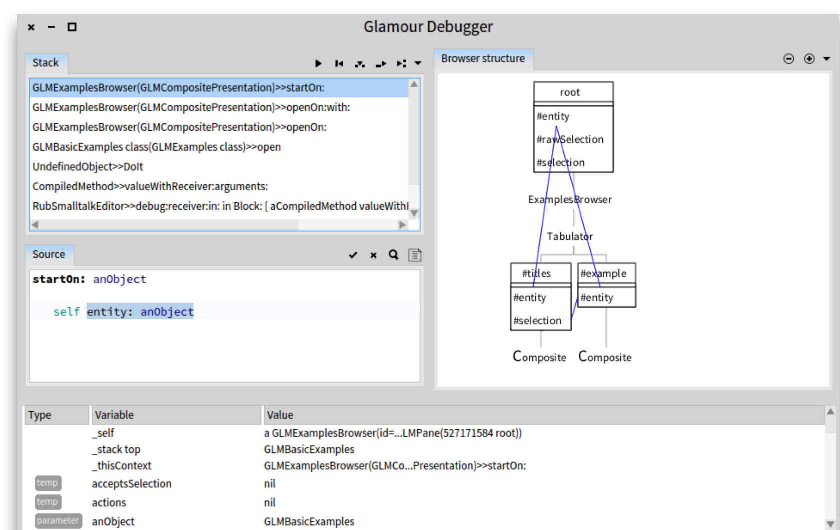
Events



SUnit



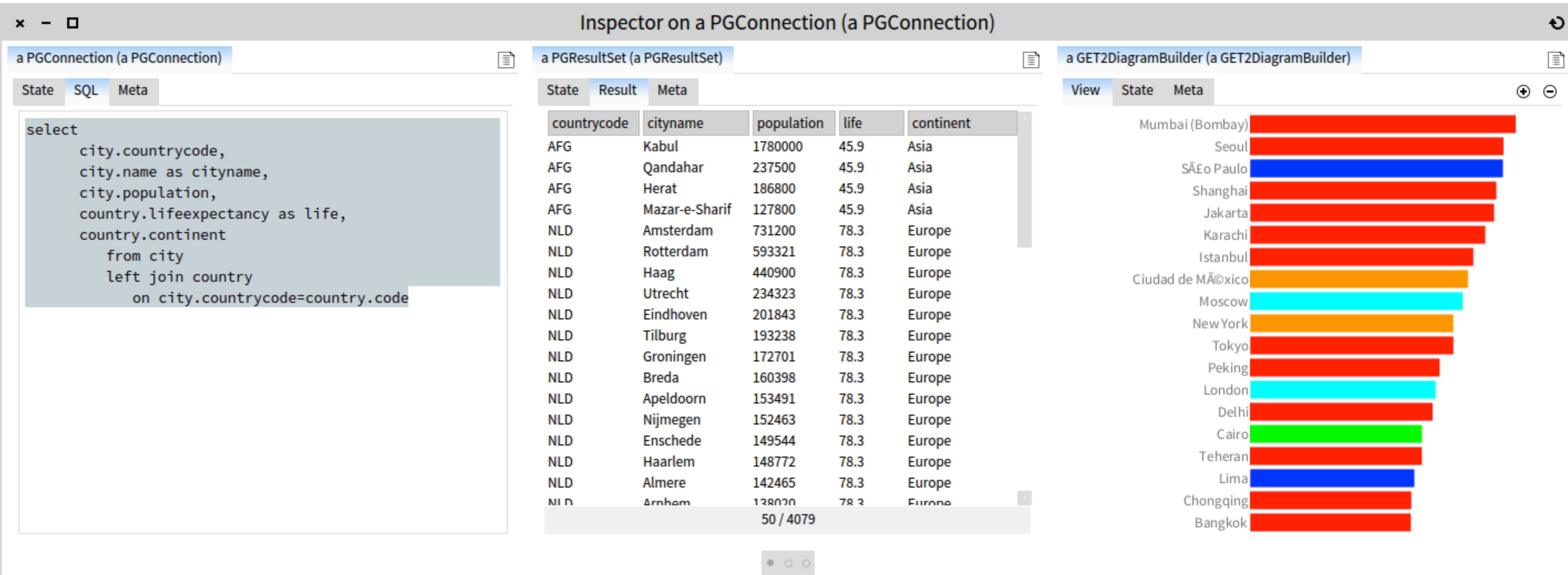
Glamour



New debuggers are cheap

	Session	Operations	View	Total
Base model	800	700	-	1500
Default Debugger	-	100	400	500
Announcements	200	50	200	450
Petit Parser	100	300	200	600
Glamour	150	100	50	300
SUnit	100	-	50	150

The Moldable Inspector



Conclusion

Current IDEs offer developers primitive support for software assessment

Developers need support for agile modeling, architectural monitoring and moldable tools