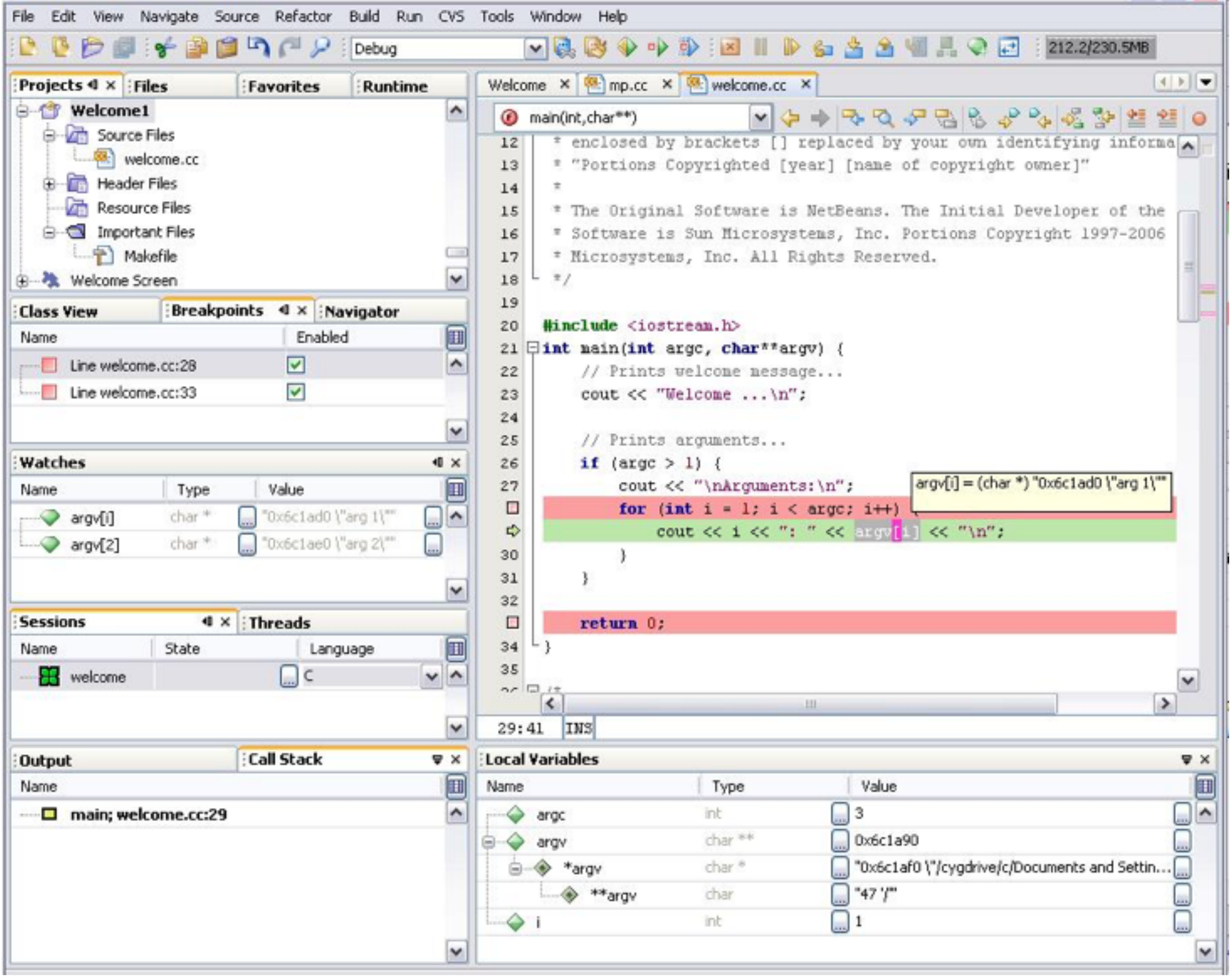
Toward a Platform for Visual Debugging

Rosario Molina, Alexandre Bergel

Pleiad Lab,
University of Chile



```
 * enclosed by brackets [] replaced by your own identifying informa
 * "Portions Copyrighted [year] [name of copyright owner]"
 *
 * The Original Software is NetBeans. The Initial Developer of the
 * Software is Sun Microsystems, Inc. Portions Copyright 1997-2006
 * Microsystems, Inc. All Rights Reserved.
 */

#include <iostream.h>
int main(int argc, char**argv) {
    // Prints welcome message...
    cout << "Welcome ...\n";

    // Prints arguments...
    if (argc > 1) {
        cout << "\nArguments:\n";
        for (int i = 1; i < argc; i++) {
            cout << i << ": " << argv[i] << "\n";
        }
    }

    return 0;
}
```

argv[i] = (char *) "0x6c1ad0 \"arg 1\""

29:41 INS

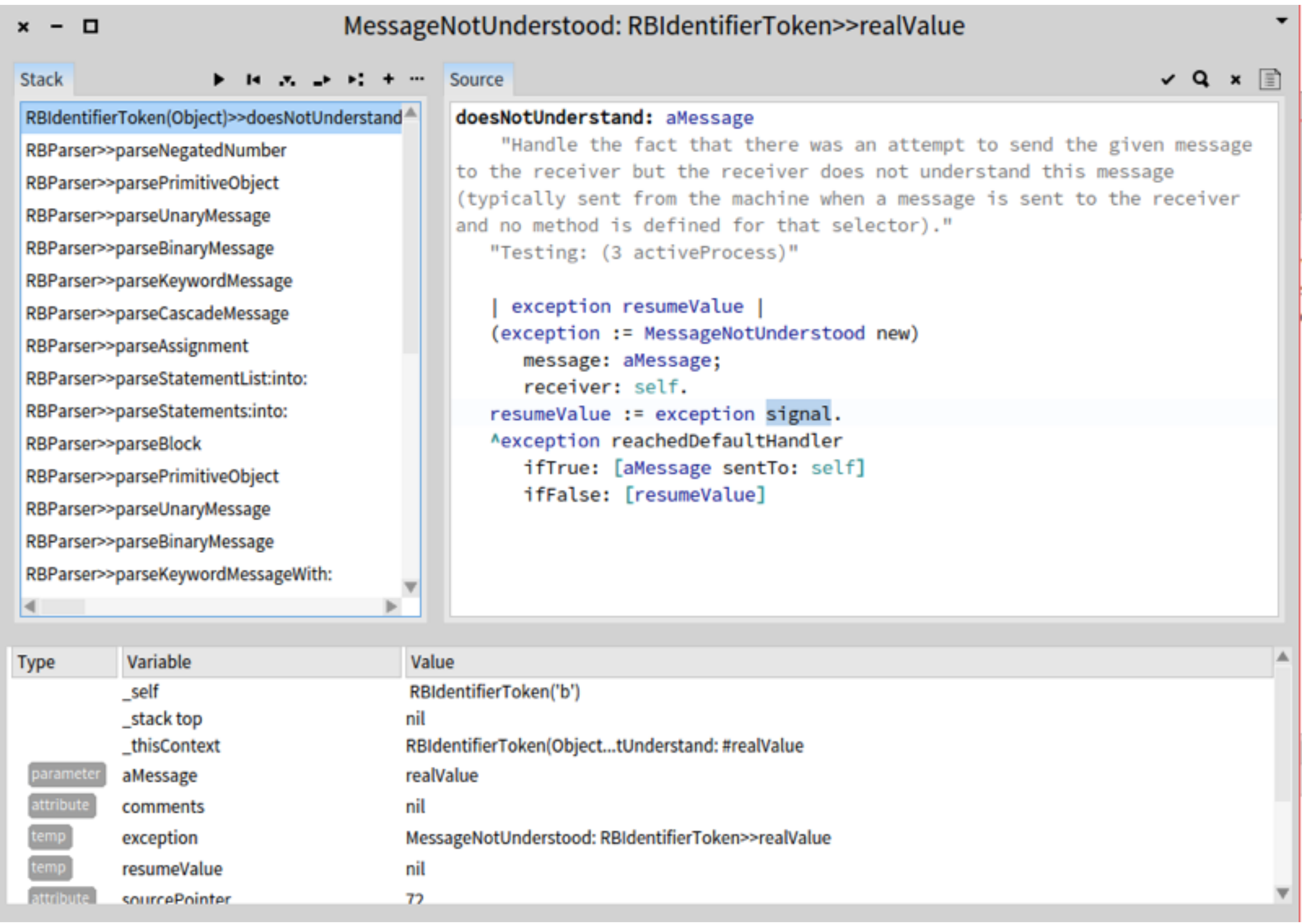
Name	Enabled
Line welcome.cc:28	☑
Line welcome.cc:33	☑

Name	Type	Value
argv[i]	char *	"0x6c1ad0 \"arg 1\""
argv[2]	char *	"0x6c1ae0 \"arg 2\""

Name	State	Language
welcome		C

Name
main; welcome.cc:29

Name	Type	Value
argc	int	3
argv	char **	0x6c1a90
*argv	char *	"0x6c1af0 \"/cygdrive/c/Documents and Settin...
**argv	char	"47 '/'"
i	int	1

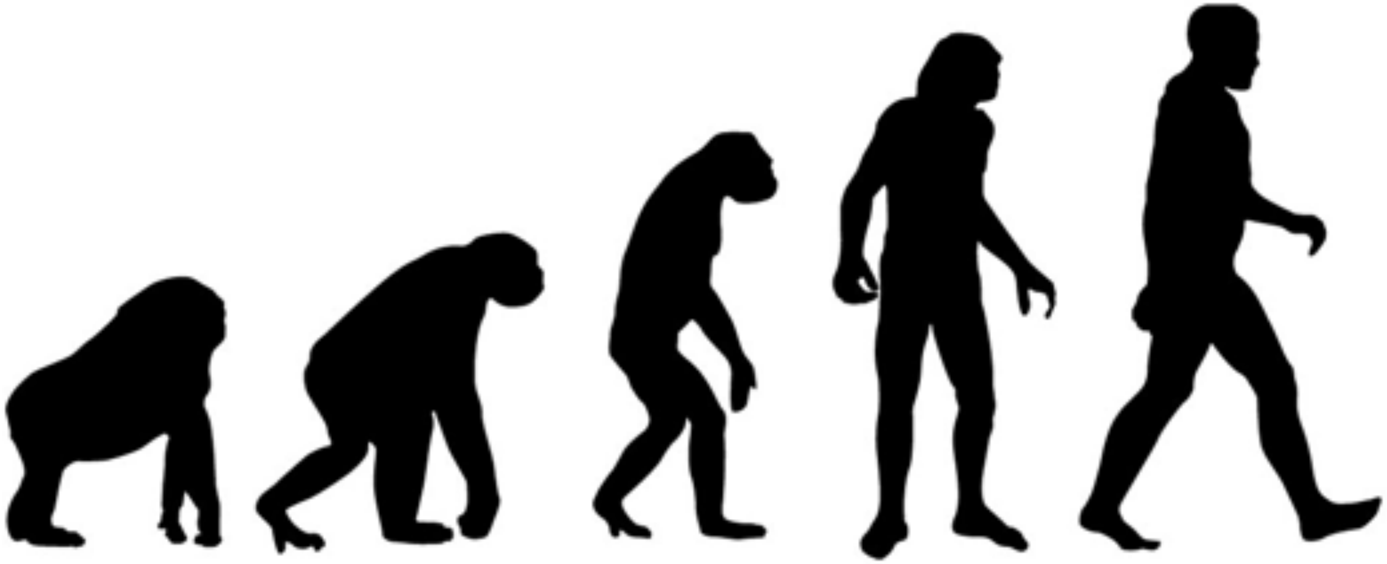


MessageNotUnderstood: RBIdentifierToken>>realValue

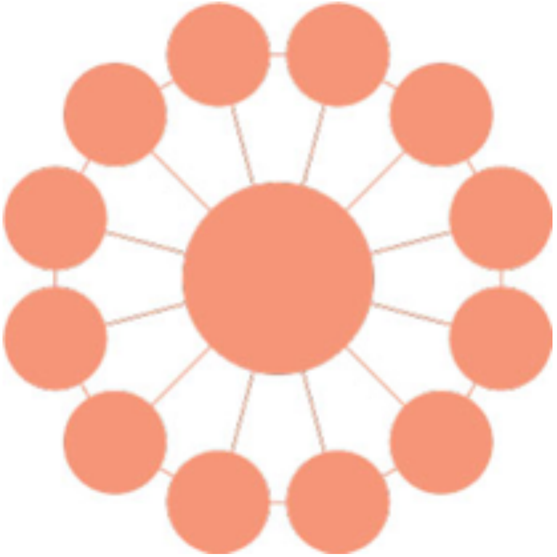
```
RBIdentifierToken(Object)>>doesNotUnderstand
RBParser>>parseNegatedNumber
RBParser>>parsePrimitiveObject
RBParser>>parseUnaryMessage
RBParser>>parseBinaryMessage
RBParser>>parseKeywordMessage
RBParser>>parseCascadeMessage
RBParser>>parseAssignment
RBParser>>parseStatementList:into:
RBParser>>parseStatements:into:
RBParser>>parseBlock
RBParser>>parsePrimitiveObject
RBParser>>parseUnaryMessage
RBParser>>parseBinaryMessage
RBParser>>parseKeywordMessageWith:
```

```
doesNotUnderstand: aMessage
    "Handle the fact that there was an attempt to send the given message
to the receiver but the receiver does not understand this message
(typically sent from the machine when a message is sent to the receiver
and no method is defined for that selector)."
   "Testing: (3 activeProcess)"

   | exception resumeValue |
   (exception := MessageNotUnderstood new)
      message: aMessage;
      receiver: self.
   resumeValue := exception signal.
   ^exception reachedDefaultHandler
      ifTrue: [aMessage sentTo: self]
      ifFalse: [resumeValue]
```

Type	Variable	Value
	_self	RBIdentifierToken('b')
	_stack top	nil
	_thisContext	RBIdentifierToken(Object...tUnderstand: #realValue
parameter	aMessage	realValue
attribute	comments	nil
temp	exception	MessageNotUnderstood: RBIdentifierToken>>realValue
temp	resumeValue	nil
attribute	sourcePointer	72

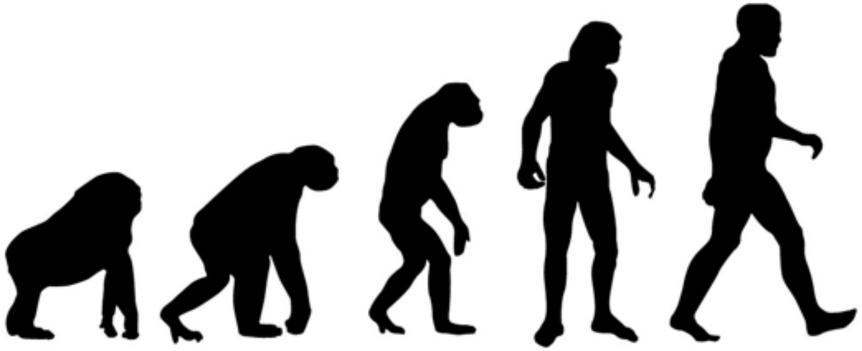


Omniscient debugging (Lienhard *et al.* 2008)

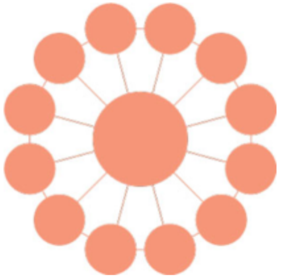


Object-centric debugging (Ressia *et al.* 2012)

Our proposal

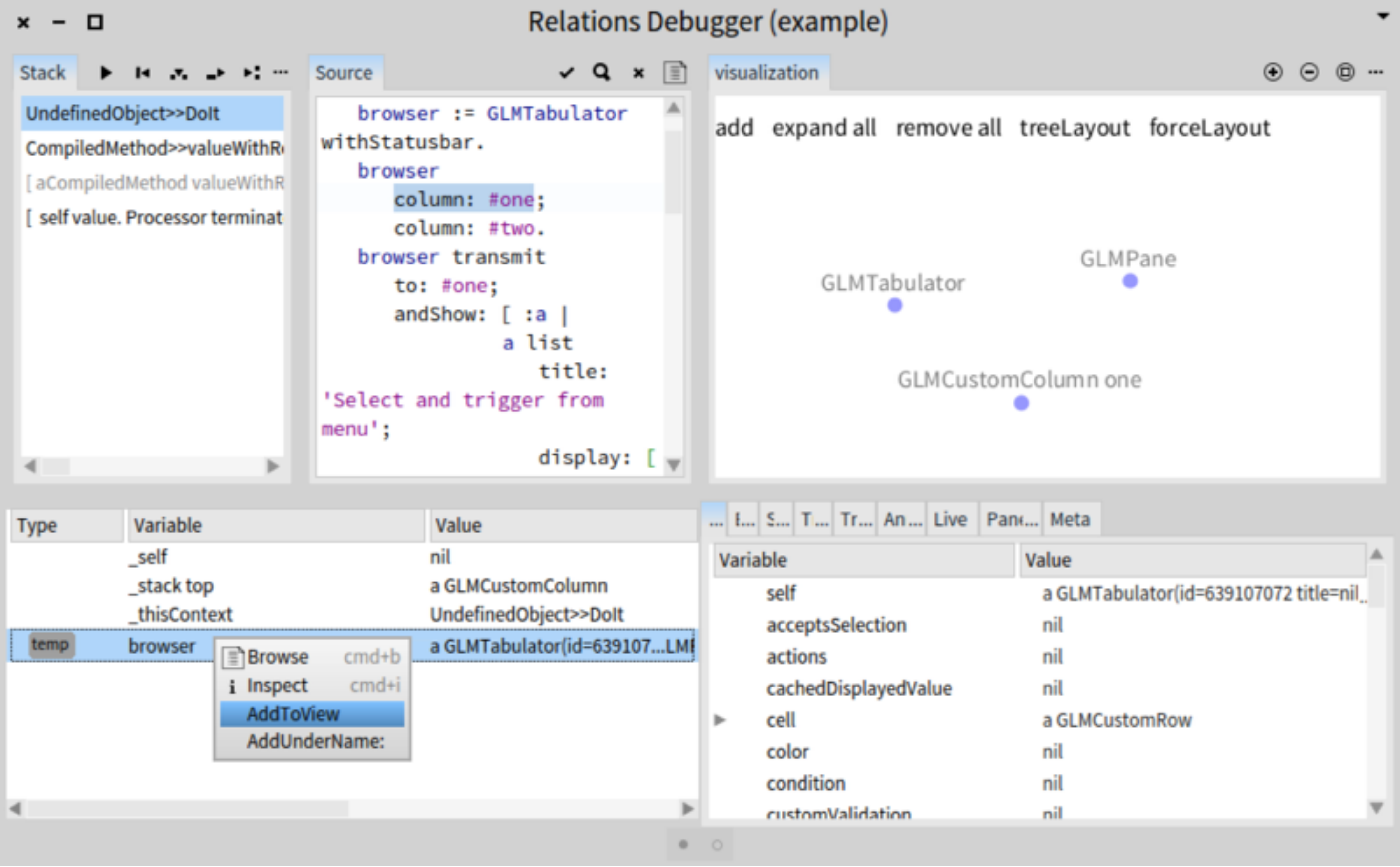


+



+

+

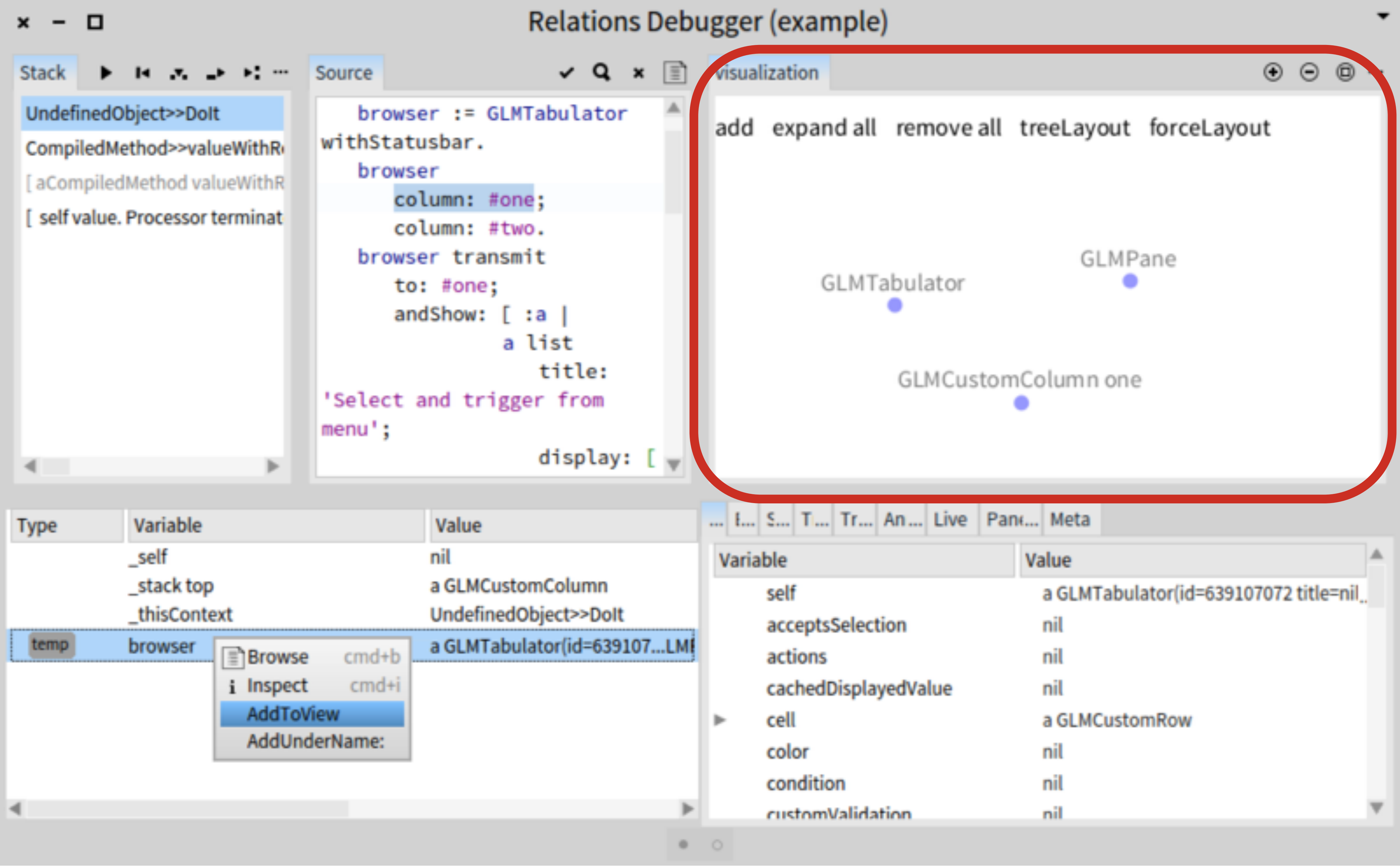


UndefinedObject>>DoIt
CompiledMethod>>valueWithR
[aCompiledMethod valueWithR
[self value. Processor terminat

```
   browser := GLMTabulator
withStatusbar.
   browser
      column: #one;
      column: #two.
   browser transmit
      to: #one;
      andShow: [ :a |
               a list
                  title:
'Select and trigger from
menu';
                  display: [
```

Type	Variable	Value
	_self	nil
	_stack top	a GLMCustomColumn
	_thisContext	UndefinedObject>>DoIt
temp	browser	a GLMTabulator(id=639107...LMI

Variable	Value
self	a GLMTabulator(id=639107072 title=nil_
acceptsSelection	nil
actions	nil
cachedDisplayedValue	nil
cell	a GLMCustomRow
color	nil
condition	nil
customValidation	nil

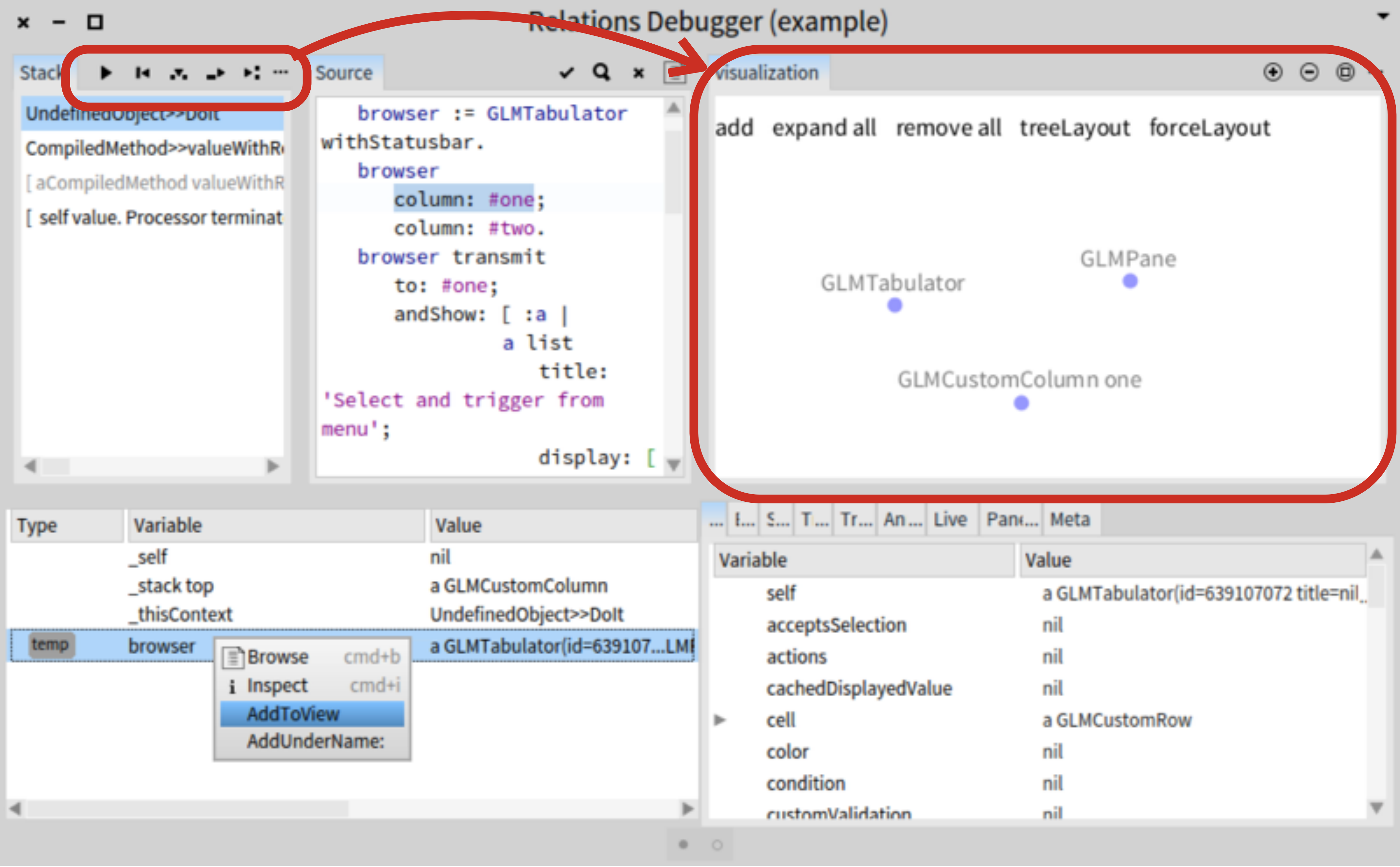


UndefinedObject>>DoIt
CompiledMethod>>valueWithR
[aCompiledMethod valueWithR
[self value. Processor terminat

```
   browser := GLMTabulator
withStatusbar.
   browser
      column: #one;
      column: #two.
   browser transmit
      to: #one;
      andShow: [ :a |
               a list
                  title:
'Select and trigger from
menu';
                  display: [
```

Type	Variable	Value
	_self	nil
	_stack top	a GLMCustomColumn
	_thisContext	UndefinedObject>>DoIt
temp	browser	a GLMTabulator(id=639107...LMI

Variable	Value
self	a GLMTabulator(id=639107072 title=nil_
acceptsSelection	nil
actions	nil
cachedDisplayedValue	nil
cell	a GLMCustomRow
color	nil
condition	nil
customValidation	nil

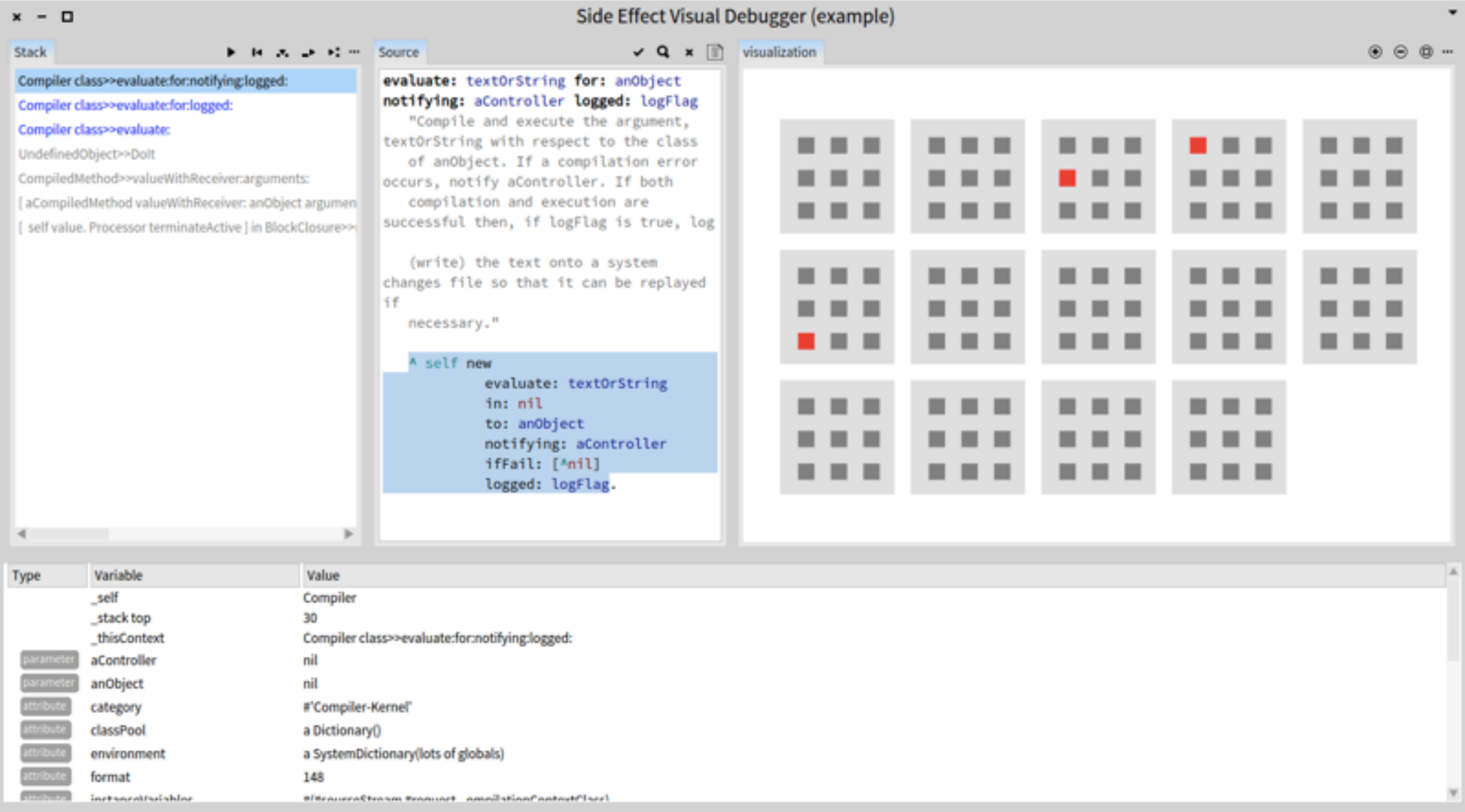


UndefinedObject>>DoIt
CompiledMethod>>valueWithR
[aCompiledMethod valueWithR
[self value. Processor terminat

```
   browser := GLMTabulator
withStatusbar.
   browser
      column: #one;
      column: #two.
   browser transmit
      to: #one;
      andShow: [ :a |
               a list
                  title:
'Select and trigger from
menu';
                  display: [
```

Type	Variable	Value
	_self	nil
	_stack top	a GLMCustomColumn
	_thisContext	UndefinedObject>>DoIt
temp	browser	a GLMTabulator(id=639107...LMI

Variable	Value
self	a GLMTabulator(id=639107072 title=nil_
acceptsSelection	nil
actions	nil
cachedDisplayedValue	nil
cell	a GLMCustomRow
color	nil
condition	nil
customValidation	nil



```
Compiler class>>evaluate:for:notifying:logged:
Compiler class>>evaluate:for:logged:
Compiler class>>evaluate:
UndefinedObject>>DoIt
CompiledMethod>>valueWithReceiver:arguments:
[ aCompiledMethod valueWithReceiver: anObject argumen
[ self value. Processor terminateActive ] in BlockClosure>>
```

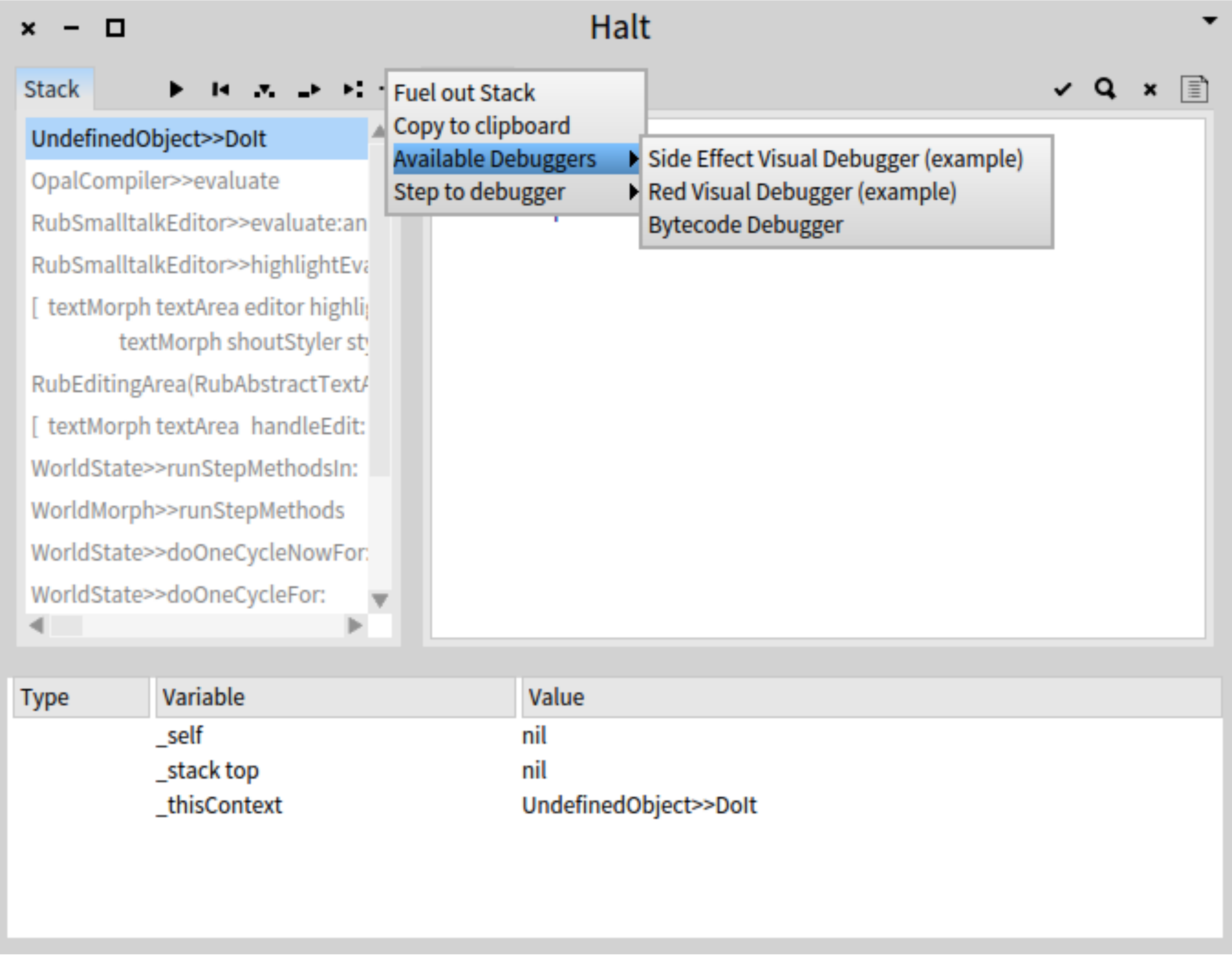
```
evaluate: textOrString for: anObject
notifying: aController logged: logFlag
   "Compile and execute the argument,
textOrString with respect to the class
   of anObject. If a compilation error
occurs, notify aController. If both
   compilation and execution are
successful then, if logFlag is true, log

   (write) the text onto a system
changes file so that it can be replayed
if
   necessary."

   ^ self new
         evaluate: textOrString
         in: nil
         to: anObject
         notifying: aController
         ifFail: [^nil]
         logged: logFlag.
```

Type	Variable	Value
	_self	Compiler
	_stack top	30
	_thisContext	Compiler class>>evaluate:for:notifying:logged:
parameter	aController	nil
parameter	anObject	nil
attribute	category	#'Compiler-Kernel'
attribute	classPool	a Dictionary()
attribute	environment	a SystemDictionary(lots of globals)
attribute	format	148
attribute	instanceVariables	#(#sourceStream #requestor #compilationContextClass)

```
AbstractVisualStackDebugger
   subclass: #VSDSideEffectDebugger
```

```
VSDDebugSession
   subclass: #VSDSideEffectDebugSession
   instanceVariableNames: 'backgroundElements
                           objectsAndValues'
```

```
VSDSideEffectDebugSession >> stepOver: aContext
  ”Executed whenever the user press the step over button”
  super stepOver: aContext.
  ...
```



```
UndefinedObject>>DoIt
OpalCompiler>>evaluate
RubSmalltalkEditor>>evaluate:an
RubSmalltalkEditor>>highlightEva
[ textMorph textArea editor highli
        textMorph shoutStyler sty
RubEditingArea(RubAbstractTextA
[ textMorph textArea  handleEdit:
WorldState>>runStepMethodsIn:
WorldMorph>>runStepMethods
WorldState>>doOneCycleNowFor:
WorldState>>doOneCycleFor:
```

Type	Variable	Value
	_self	nil
	_stack top	nil
	_thisContext	UndefinedObject>>DoIt

Is it really worth researching on this?

Is it really worth researching on this?

J. Sillito, G. C. Murphy, K. De Volder, “Questions programmers ask during software evolution tasks”, SIGSOFT ’06

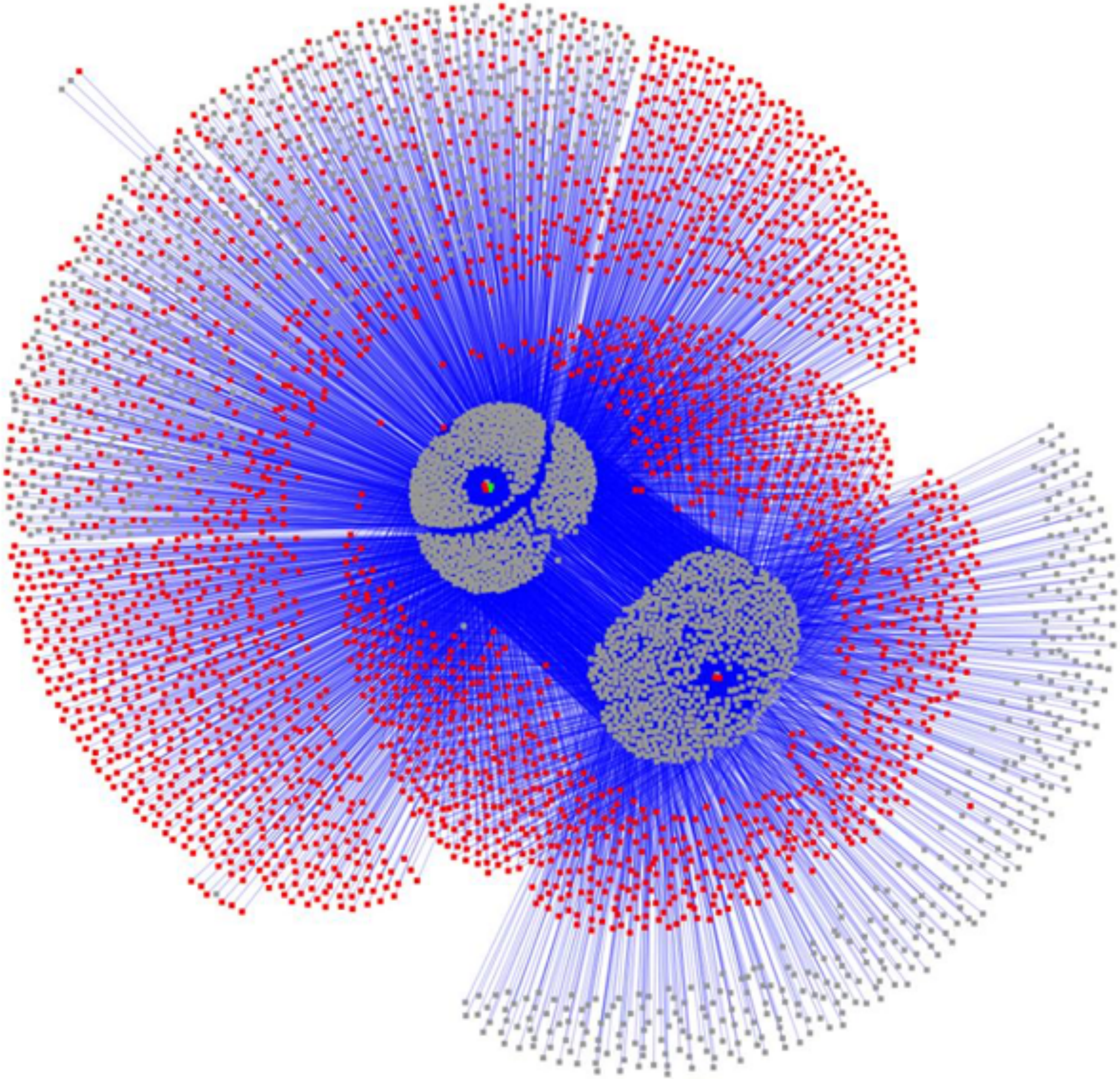
44 kind of questions that programmers ask themselves when they perform a change task on a code base:

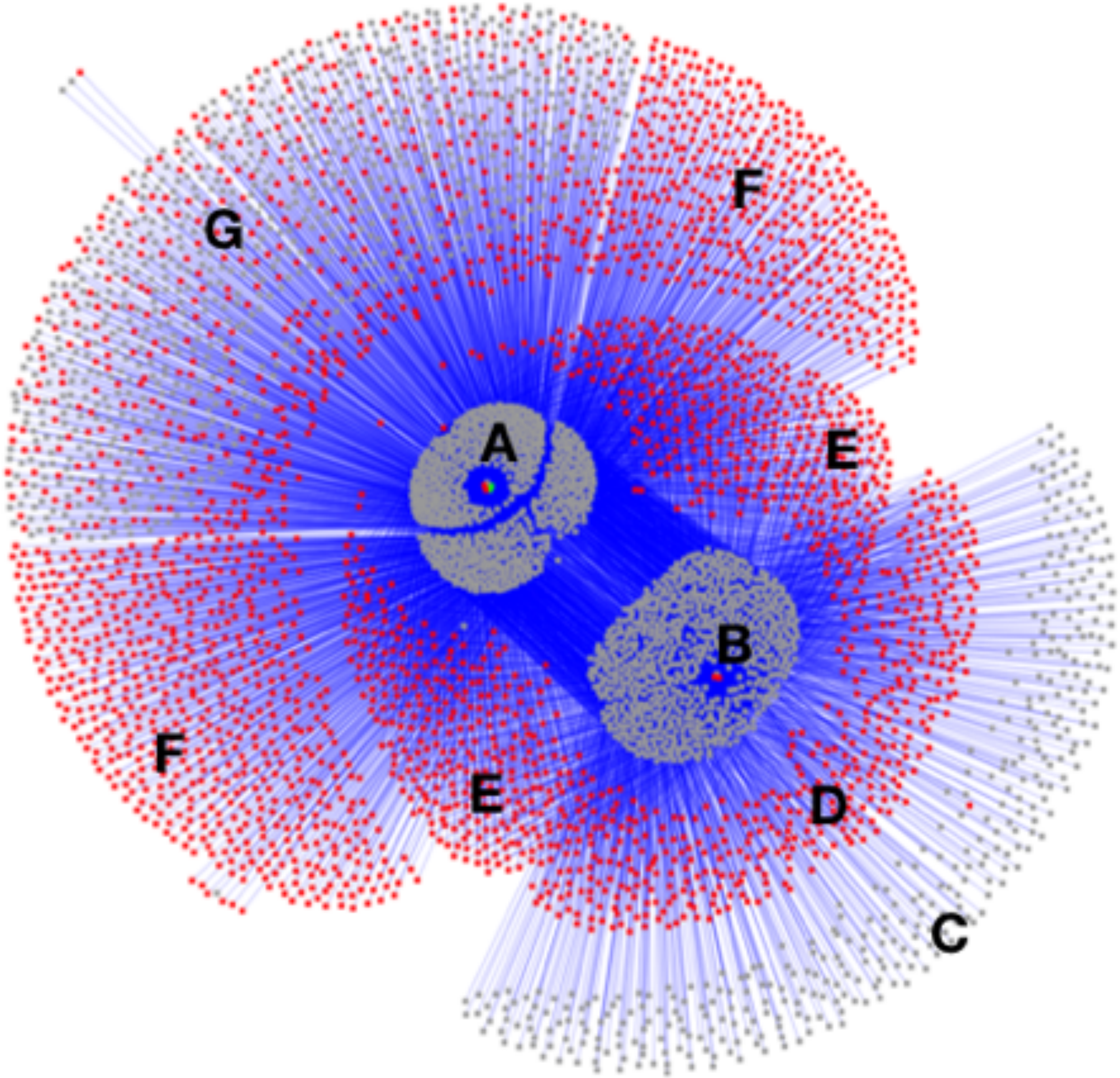
Where is this variable or data structure being accessed?

What data can we access from this object?

What data is being modified in this code?

How are these types or objects related?





A - Trachel
B - Roassal
C - Instances of RTGroup
D - Glamour related instances
E - Event related (Announcer)
F - Colors
G - Points

Look for VisualStackDebugger on Smalltalkhub