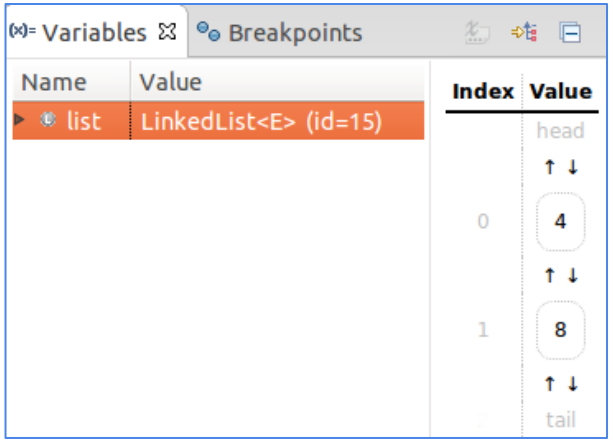
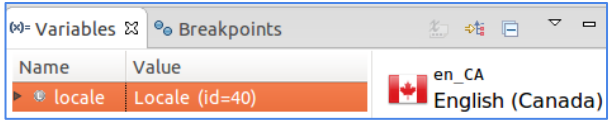
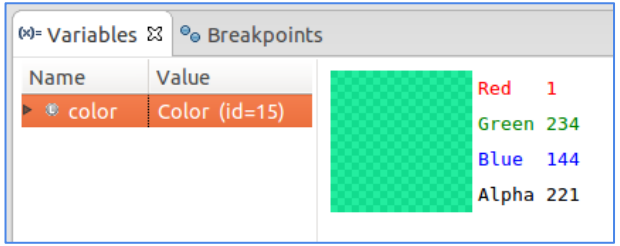
Templated visualization of object state with Vebugger







Daniel Rozenberg
Ivan Beschastnikh

Computer Science
University of British Columbia

Motivation

Inspecting program state is key to software engineering

For example, some common steps in debugging are:

- Step through code
- Inspect state
- Make changes

Motivation

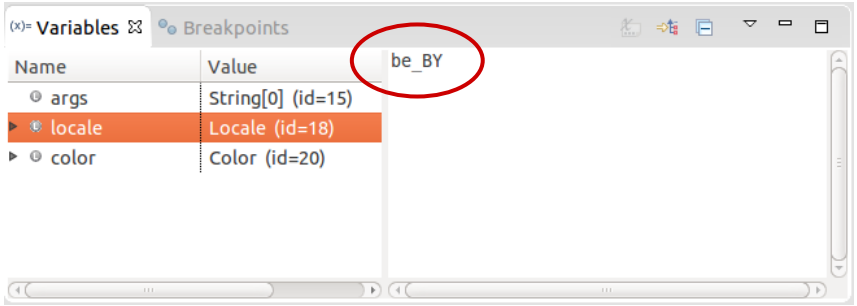
Inspecting program state is key to software engineering

For example, some common steps in debugging are:

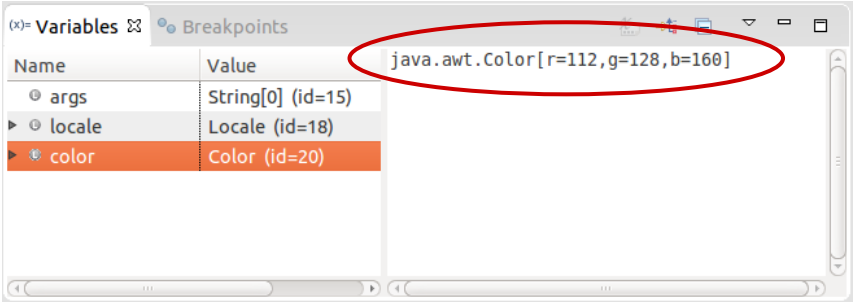
- Step through code
- **Inspect state**
- Make changes

State inspection: still hard

Mind the abstraction gap

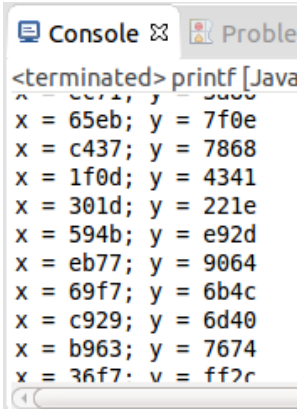


Which language and country are represented here?



What color is this?

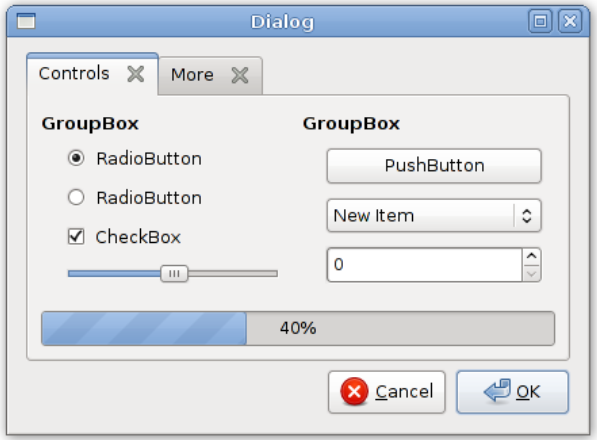
Established approaches to state inspection



printf

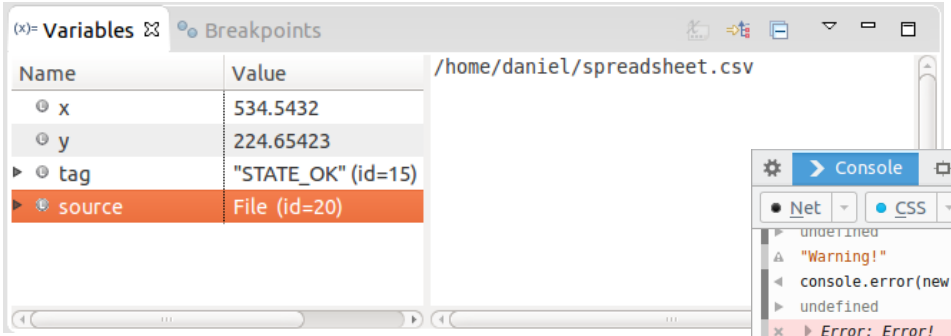
```
daniel@Anders:~$ gdb x_and_y
GNU gdb (Ubuntu 7.7-0ubuntu3.1) 7.7
Reading symbols from x_and_y...done.
(gdb) break x_and_y.c:6
Breakpoint 1 at 0x4004f8: file x_and_y.c, line 6.
(gdb) run
Starting program: /home/daniel/x_and_y

Breakpoint 1, main () at x_and_y.c:6
6               y = 2;
(gdb) p x
$1 = 5
(gdb) p y
$2 = 0
(gdb) step 1
8               return 0;
(gdb) p x
$3 = 5
(gdb) p y
$4 = 2
(gdb) c
Continuing.
[Inferior 1 (process 8229) exited normally]
(gdb) quit
daniel@Anders:~$
```

gdb

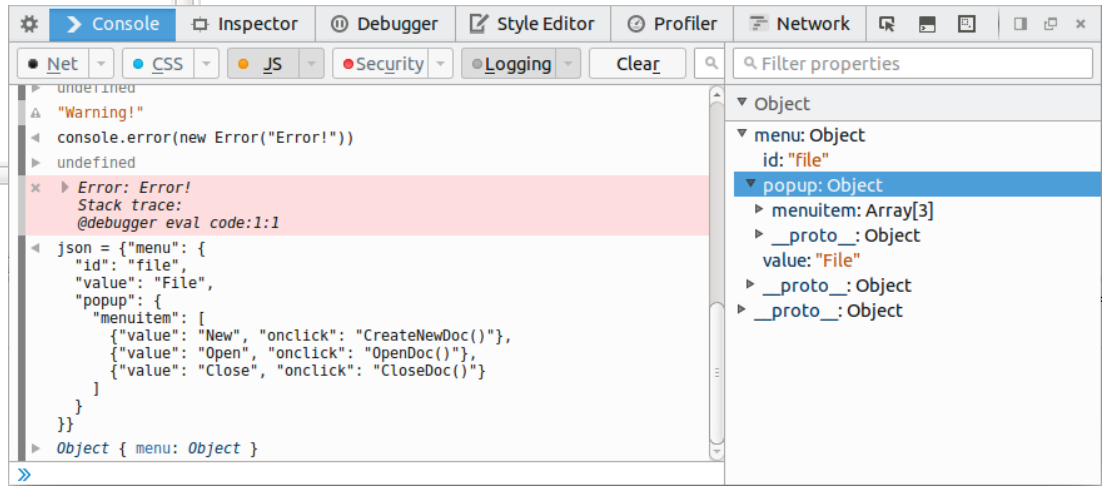


inspecting GUI directly

Recent approaches to state inspection

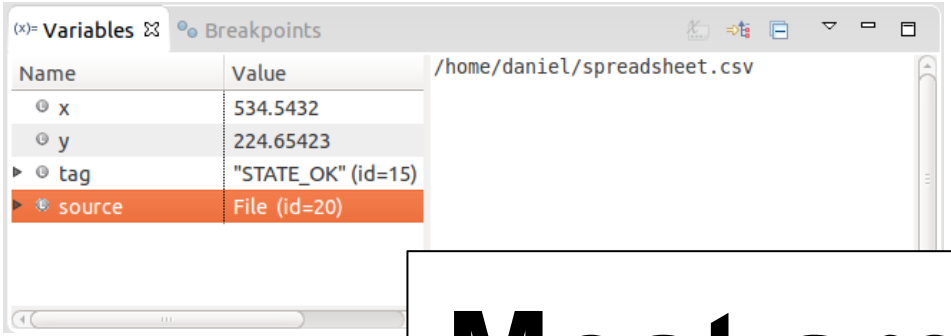


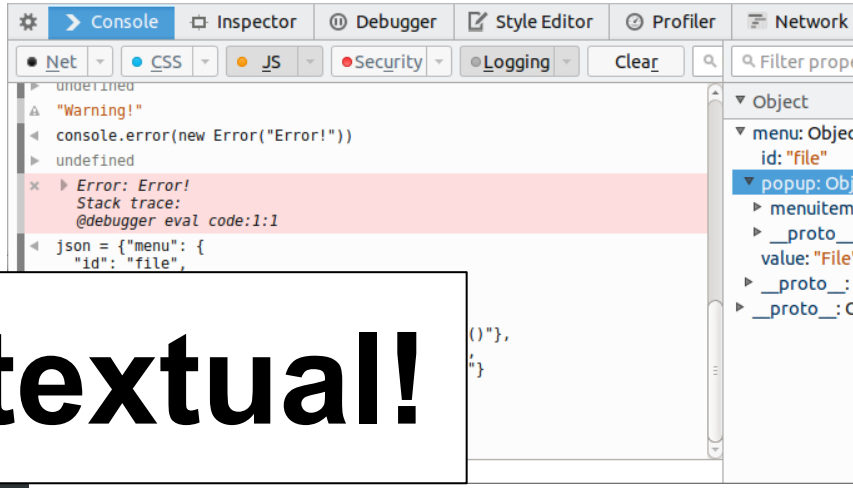
IDE integrated debugger



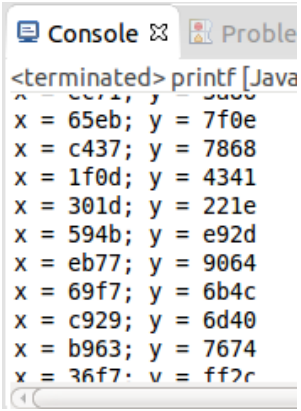
Browser-based development tool

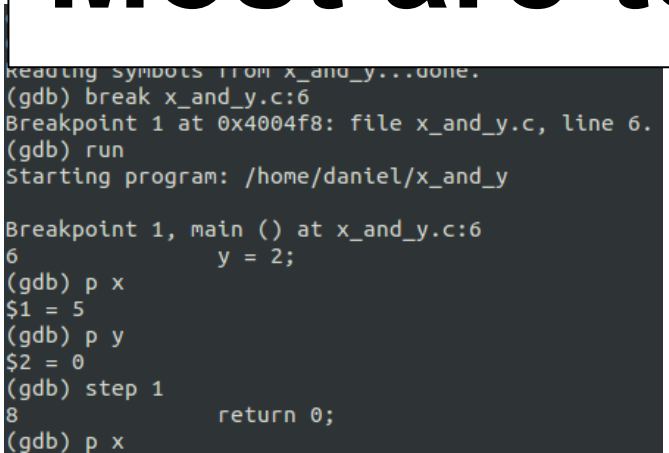
Approaches to state inspection

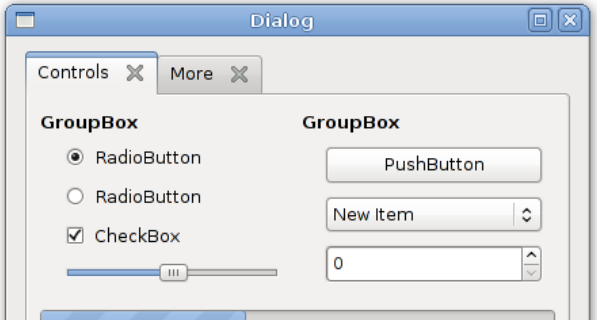




Most are textual!



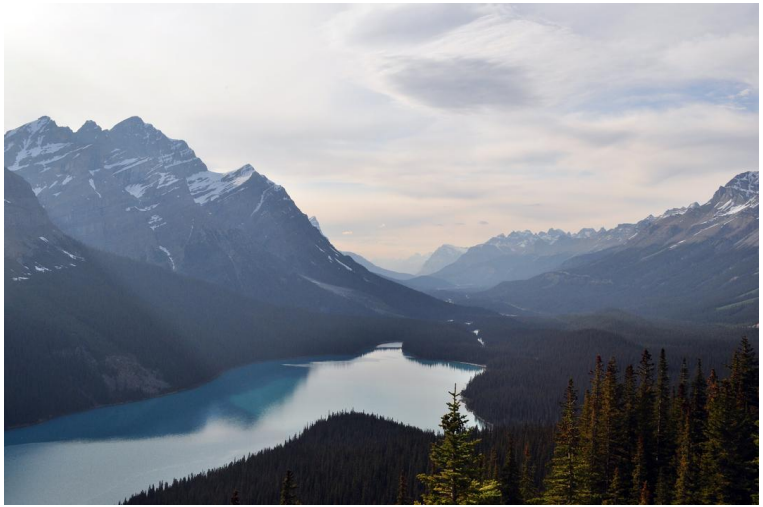




Text has its limits

UI elements, media, data structures, etc.

```
Image image = ImageIO.read(new File("/home/daniel/pyeto_lake.jpeg"));
```



>

```
BufferedImage@60f4ffd8: type = 5
ColorModel: #pixelBits = 24
numComponents = 3 color space =
java.awt.color.
ICC_ColorSpace@74c8efa2
transparency = 1 has alpha =
false isAlphaPre = false
ByteInterleavedRaster: width =
4288 height = 2848
#numDataElements 3 dataOff[0] =
2
```

Our goal

Mitigate the abstraction gap by augmenting the debugger's textual representation of object state with a visual representation.

Vebugger

Design criteria

- Typed visualizations
  - Visualizations should be distinguished by classes
- Extensibility through templates
  - Easy to create templates
- IDE integration
  - Developers expect tools to be integrated into IDE
- **Do no harm**
  - Revert to default behavior on any failure

Vebugger in a nutshell

- Typed visualizations
  - Uses Java types to determine which template to use
- Extensibility through templates
  - Uses HTML+CSS
- IDE integration
  - Integrates into Eclipse's "variable view" panel
- **Do no harm**
  - Displays the .toString() value when template missing

Demo time!

Future work

- Context-specific templates
- Navigation through visualizations
- Scalable visualizations
- Usability/viability user study
- Automating the template creation process
- Animating state transitions

Context-specific templates

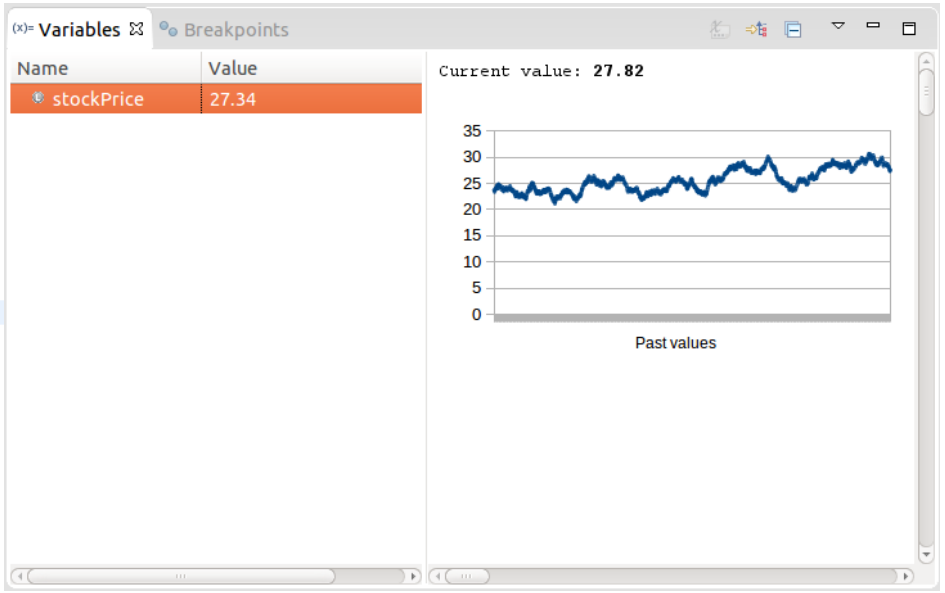
StockTrackerTimerTask.java

```
import java.util.TimerTask;

public class StockTrackerTimerTask extends TimerTask {

    private final StockPriceSource source;
    private final String symbol;

    private double tickerPrice;

    public StockTrackerTimerTask(StockPriceSource source, String symbol) {
        this.source = source;
        this.symbol = symbol;
    }

    @Override
    public void run() {
        tickerPrice = source.getCurrentStockPriceBySymbol(symbol);
    }

    public double getTickerPrice() {
        return tickerPrice;
    }

}
```

Context-specific templates

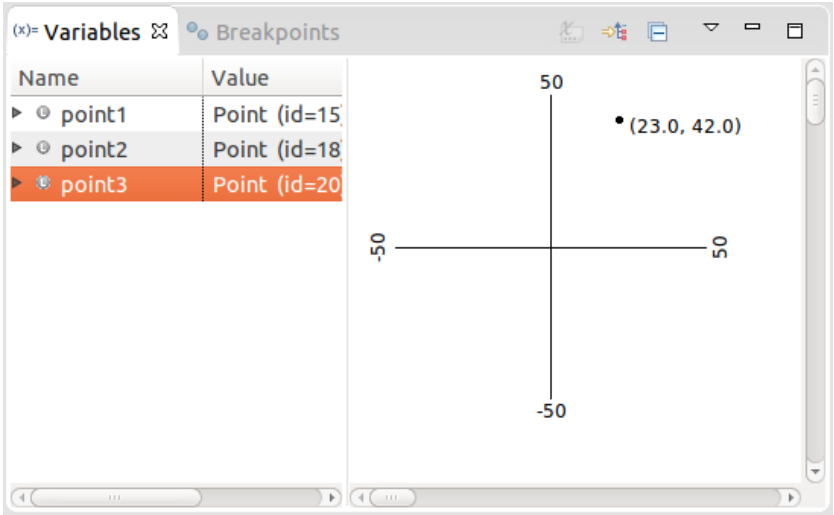
StockTrackerTimerTask.java

```
import java.util.TimerTask;

public class StockTrackerTimerTask extends TimerTask {

    private final StockPriceSource source;
    private final String symbol;

    private double tickerPrice;

    public StockTrackerTimerTask(StockPriceSource source, String symbol) {
        this.source = source;
        this.symbol = symbol;
    }

    @Override
    public void run() {
        tickerPrice = source.getCurrentStockPriceBySymbol(symbol);
    }

    public double getTickerPrice() {
        return tickerPrice;
    }

}
```



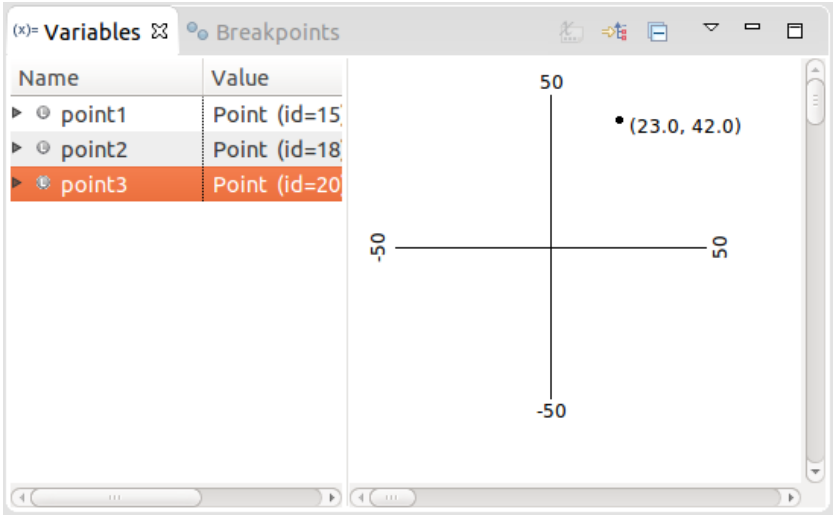
Beck et al. *VISSOFT*, 2013

Context-specific templates

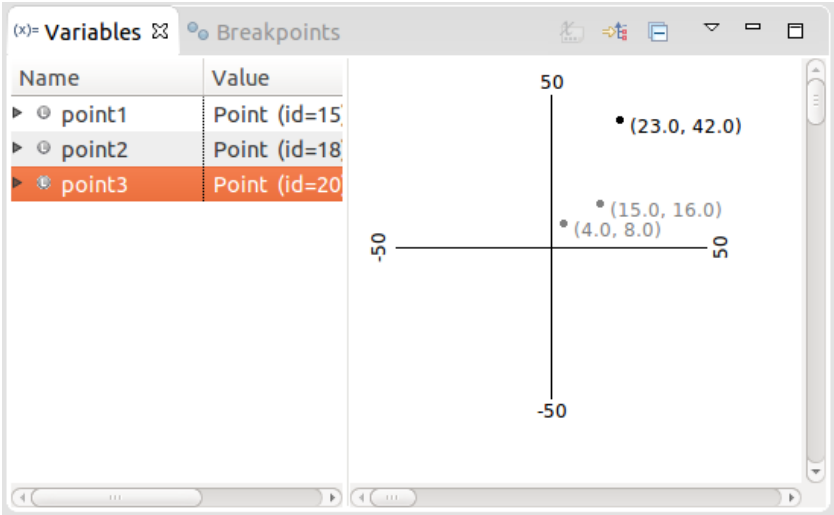


Without context

Context-specific templates



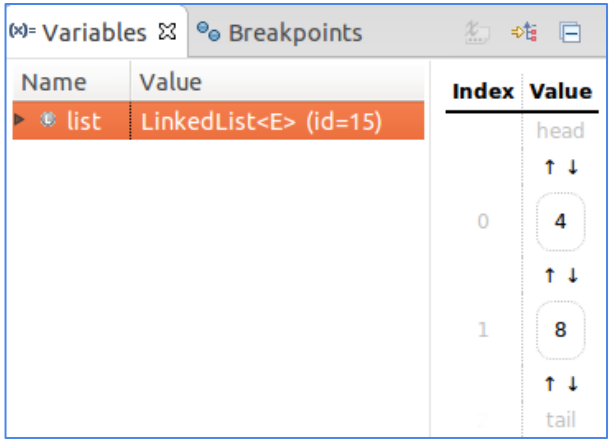
Without context

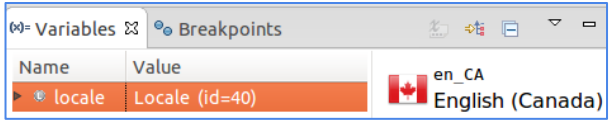


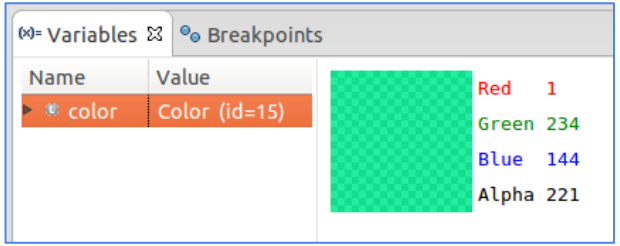
With context

Contributions

- Articulated design criteria for tools that expose object state for debugging purposes
- Built Vebugger, a framework for visualizing type-specific object state in an IDE







Vebugger is free software!
https://github.com/daniboy/vebugger

Backup slides

Related works

Other tools that show visual state

- T. D. Hendrix, J. H. Cross II, and L. A. Barowski. An extensible framework for providing dynamic data structure visualizations in a lightweight IDE. *ACM SIGCSE Bulletin*, 36(1):387–391, 2004
- C. Demetrescu and I. Finocchi. A data-driven graphical toolkit for software visualization. In *VISSOFT*, 2006
- B. Alsallakh, P. Bodesinsky, S. Miksch, and D. Nasseri. Visualizing Arrays in the Eclipse Java IDE. In *CSMR*, 2012

Exposing context-sensitive state

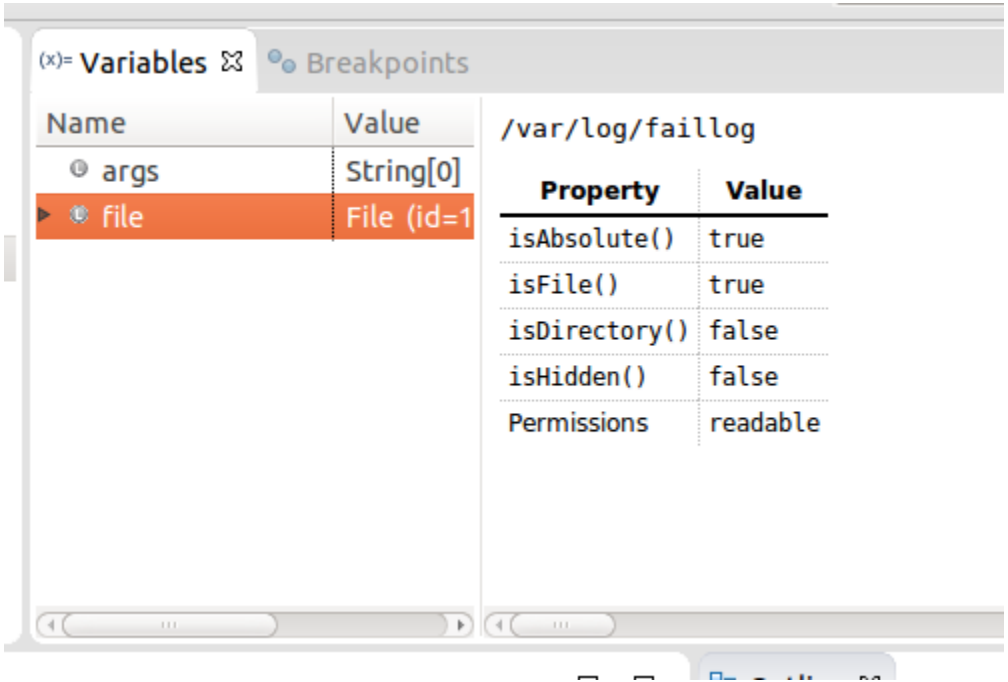
- D. A. Mellis. Tangible code. Master's thesis, Interaction Design Institute Ivrea, 2006
- F. Beck, F. Hollerich, S. Diehl, and D. Weiskopf. Visual monitoring of numeric variables embedded in source code. In *VISSOFT*, 2013
- A. J. Ko and B. A. Myers. Debugging reinvented. In *ICSE*, 2008

User studies of Vebugger

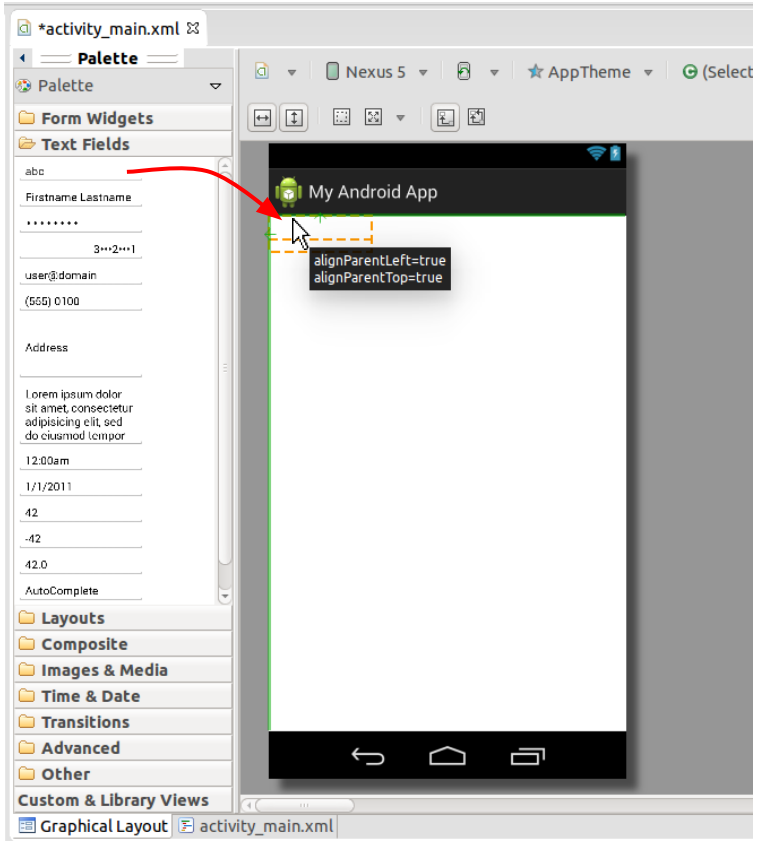
Interesting research questions:

- Which types can be visualized?
- Does comprehension of object state increase for developers who use Vebugger?

Automatic custom templates



WYSIWYG Editor for templates



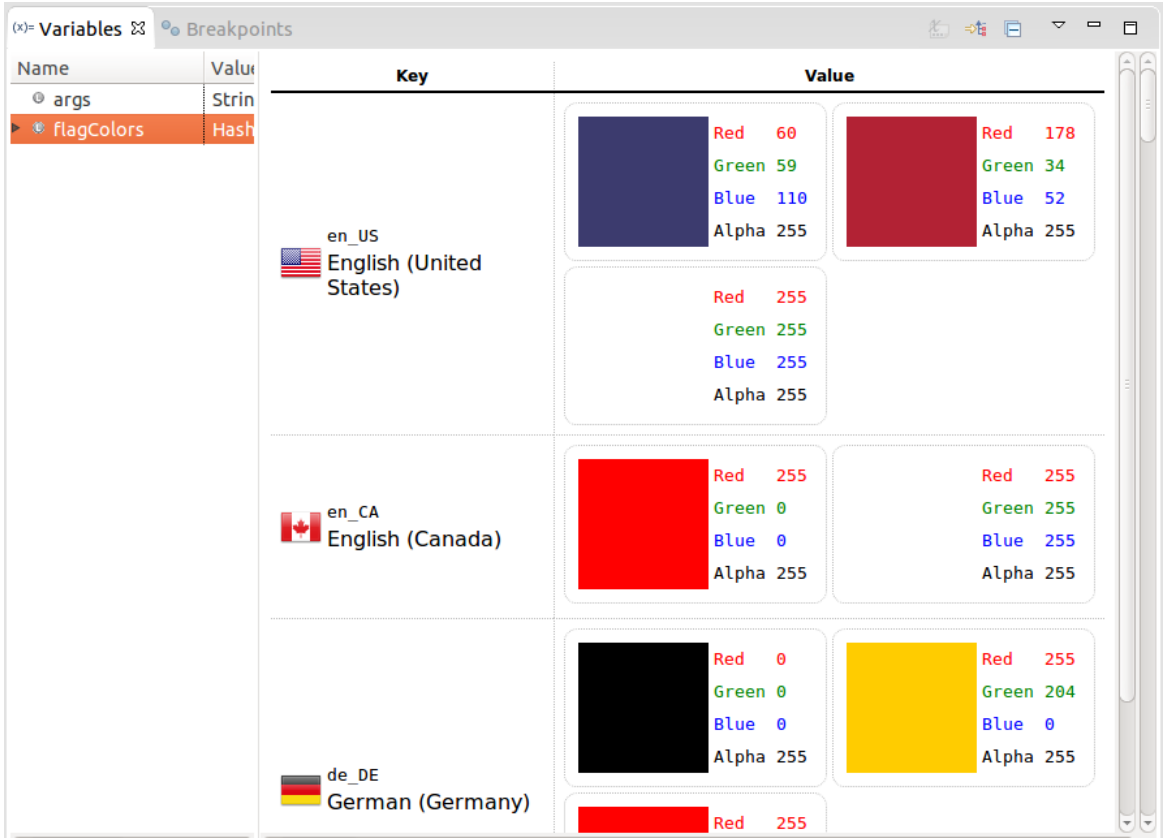
Limitations

- Heterogeneity — too many classes, too many contexts! How to streamline the template creation process to become a part of the debugging process?
- Scalability — exposing big-data without overwhelming the user or missing out on details. An open problem in Information Visualization.

Backup backup slides

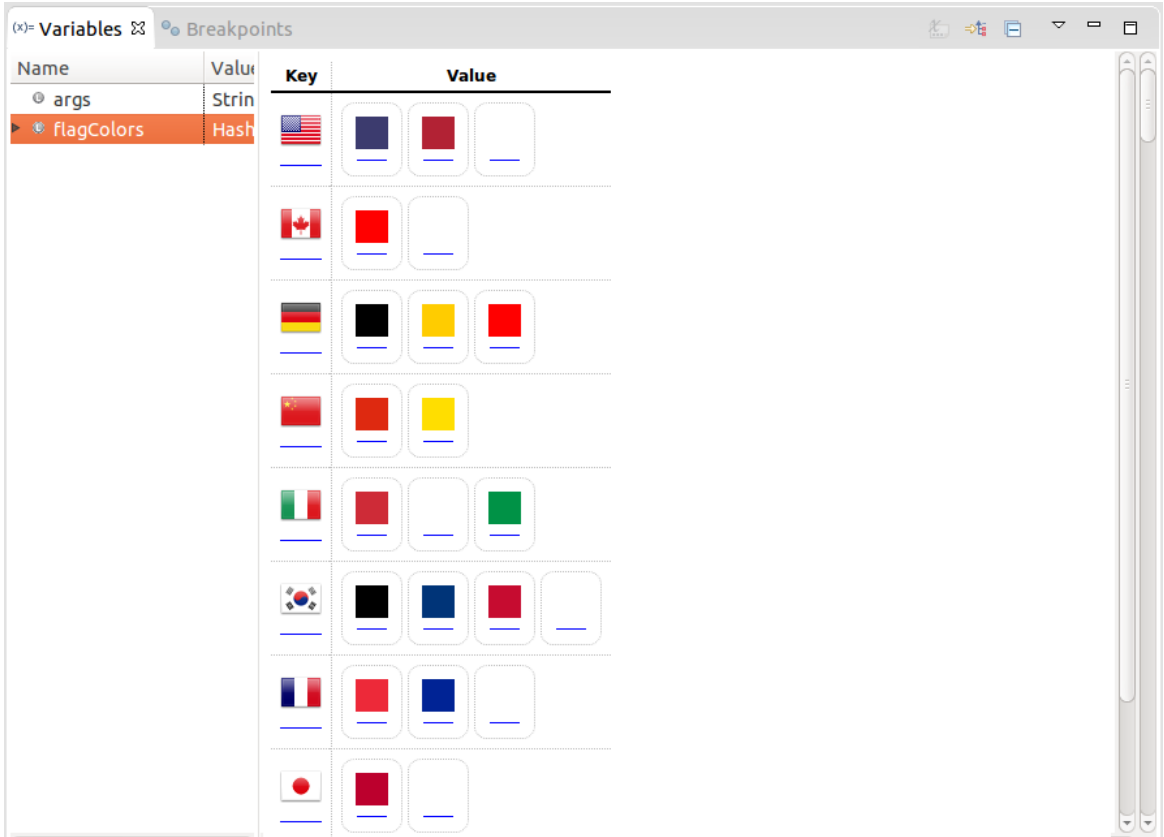
Navigation with the aid of visualizations

Map<Locale, Set<Color>> flagColors = …



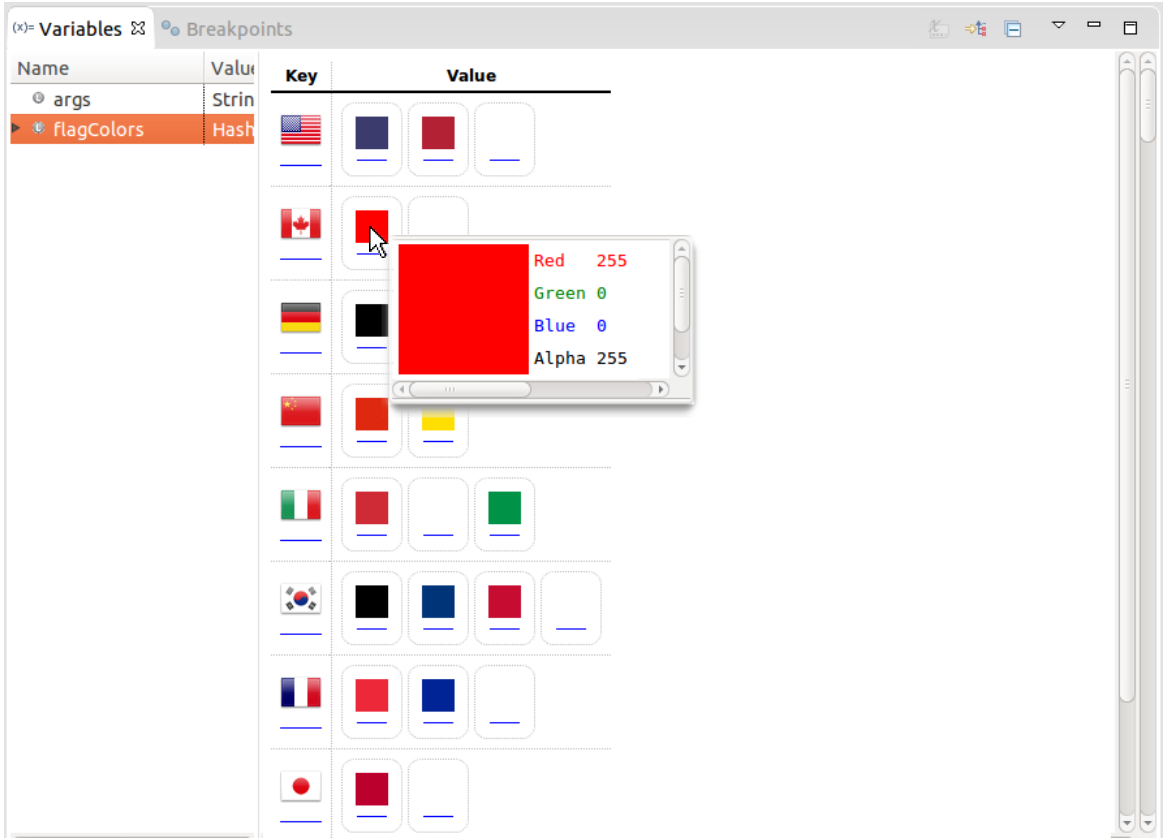
Navigation with the aid of visualizations

Map<Locale, Set<Color>> flagColors = …

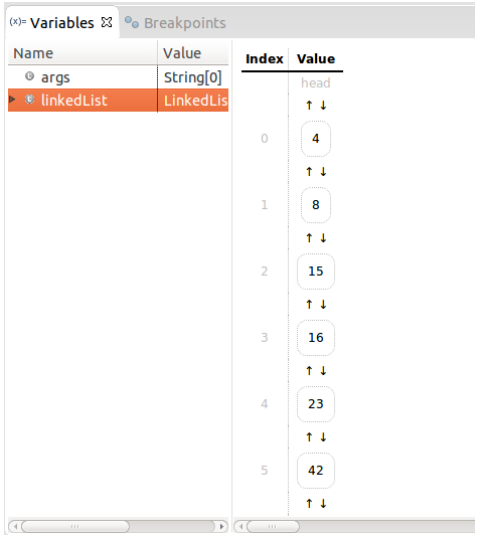


Navigation with the aid of visualizations

Map<Locale, Set<Color>> flagColors = …



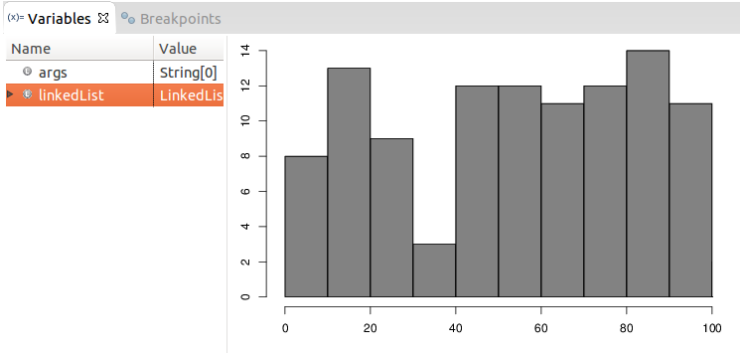
Scaling visualizations



Current implementation

PLACEHOLDER

“Fish-eye” visualization



Values histogram