Templated visualization of object state with Vebugger

Daniel Rozenberg
Ivan Beschastnikh

Computer Science
University of British Columbia
State inspection

Mind the abstraction gap
State inspection — still hard!
Mind the abstraction gap

Which **language** and **country** are represented here?

What **color** is this?
What developers use today
What developers use today

Predominantly textual!
Our goal
Mitigate abstraction gap in state inspection

Vebugger

Textual + Visual
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
Design criteria — Vebugger

● 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

Context could refer to:

- Program domain
- Runtime environment
- Developer’s task
- Operating system state
- etc…
Context (domain) templates

```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;
    }
}
```

Stock market application
Context *(domain)* templates

```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;
    }
}
```

Stock market application
Context (domain) templates

```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;
    }
}
```

Stock market application

Beck et al. VISSOFT, 2013
Context (domain) templates
Context \( (\text{domain}) \) templates

How to select a domain template?
- Manual user selection \textit{simplest solution}
- Infer from variable names \textit{static analysis}
- Dynamic object inspection \textit{dynamic analysis}
Context *(runtime)* templates

Without context
Context (*runtime*) templates

Without context

With context
Conclusion

Motivation: abstraction gap in state inspection

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

Vebugger is free software!
https://github.com/daniboy/vebugger
Backup slides
Related works

Other tools that show visual state


Exposing context-sensitive state

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.
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 = ...