Meta-tooling

Andrei CHIS
SCG, University of Bern

Software systems are complex

Software systems model
dynamic and complex fields

Software systems combine multiple programming languages and technologies

Contextual problems require contextual solutions

Software systems are highly heterogeneous

Impossible to predict all developers questions

Generic tools vs. Meta-tools

Generic tools

Answer a predefined number of questions

Answer a predefined number of questions

questions are difficult to predict

Difficult to compose and reuse

Meta-tools

Environment for building custom tools

Custom problems require custom tools

Determining the relevant questions falls on the developer and not on the tool provider

Building custom tools should be ‘cheap’

Minutes instead of hours or days

Custom tools should be composable

Custom tools should be reusable

The Glamorous Debugger

View

Model

View

Model

GTDebugger

GTDebuggerSession
GTDebuggerContex

GTDebuggerSession

```
stepInto: selectedContext
        context := process step: selectedContext.
        process stepToSendOrReturn.
```

.

.

.

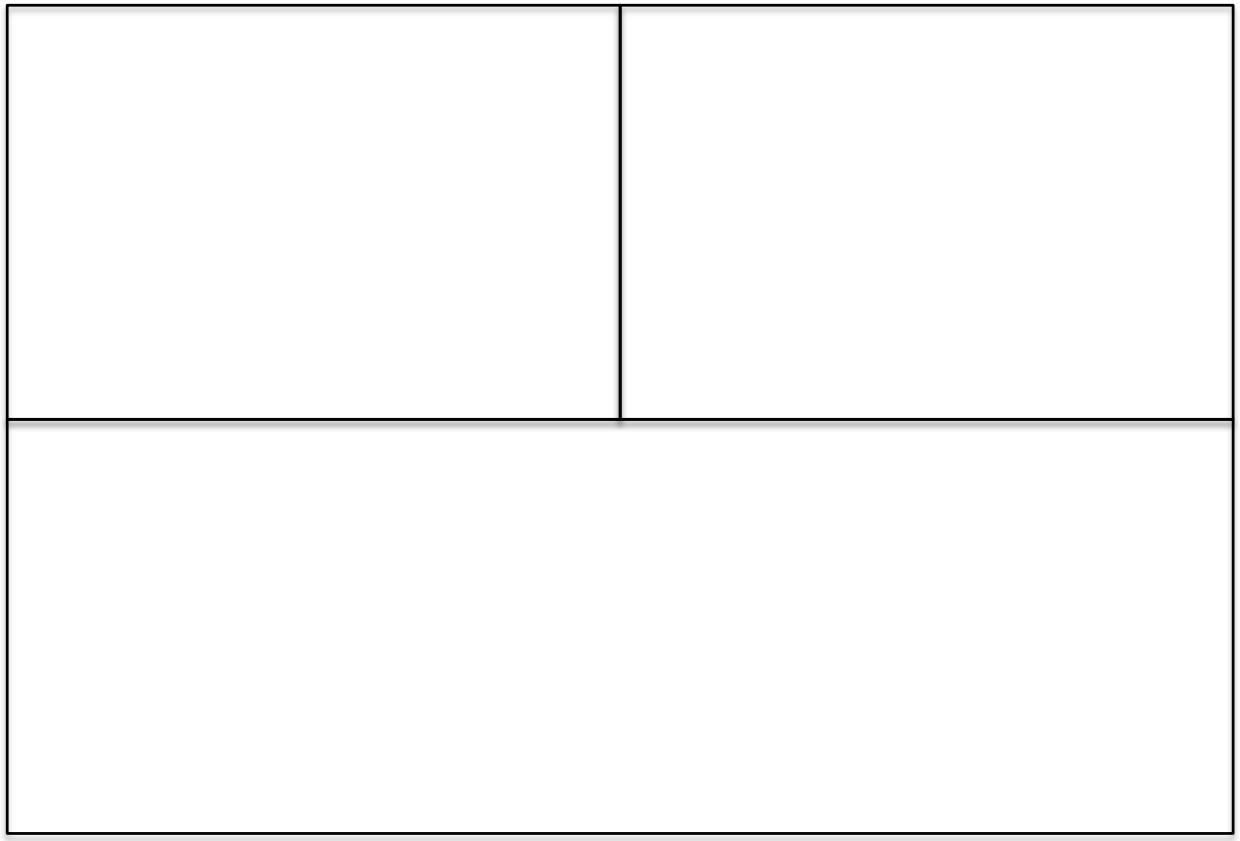
GTDebuggerContex

```
source
    ^ method holdsTempNames
        ifTrue: [method getSource ]
        ifFalse: [ methodNode sourceText ]
```

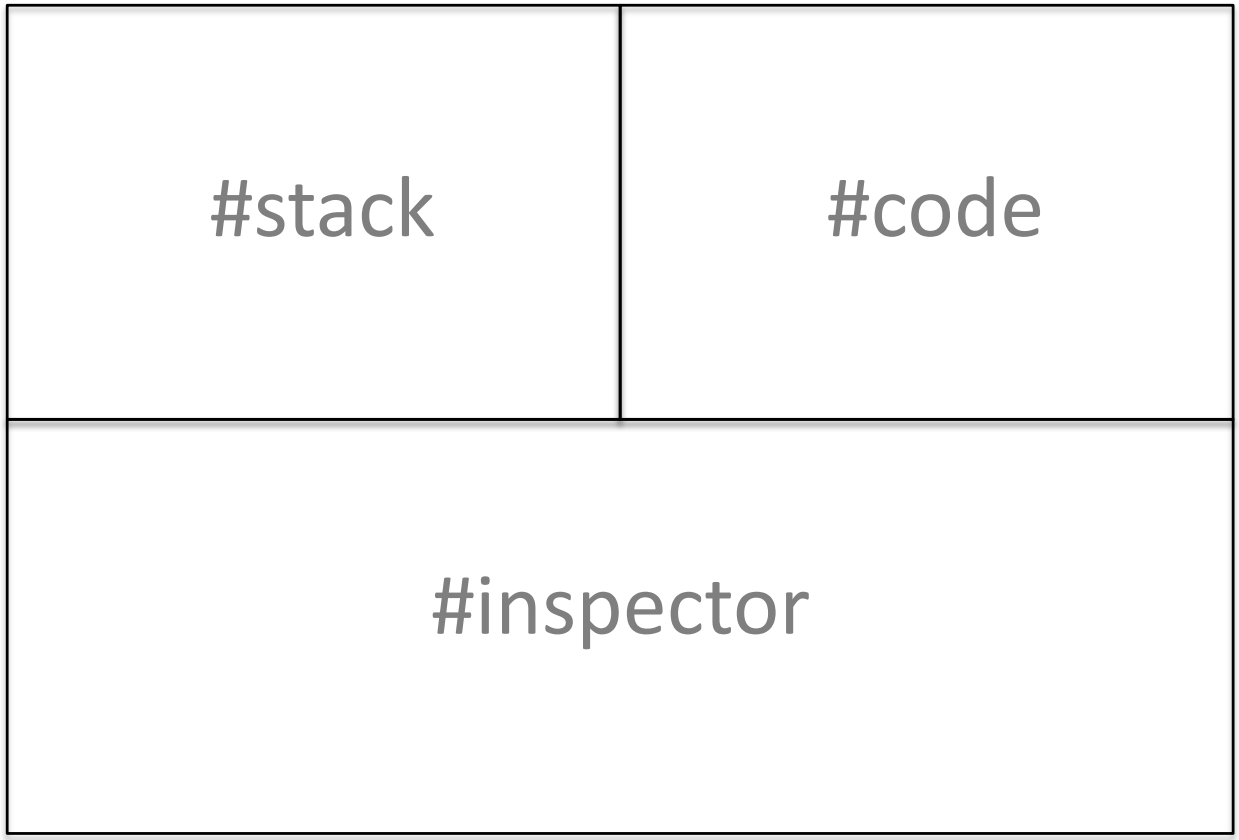
.
.
.

GTDebugger

GTDebugger



GTDebugger



GTDebugger

```
browser
	row: [ :row |
		row
			column: #stack;
			column: #code ];
	row: #inspector.
```

GTDebugger

```
browser transmit
	to: #stack;
	andShow: [ :composite | self stackIn: composite ].
```

GTDebugger

```
browser transmit
	from: #stack port: #context;
	to: #code;
	andShow: [ :composite | self methodCodeIn: composite ].
```

GTDebugger

```
browser transmit
    from: #stack port: #context;
    to: #inspector;
    andShow: [ :composite | self inspectorForContextAndSelfIn: composite ]
```

Easy to build?

Easy to build?

125 lines of code

Easy to build?

125 lines of code

412 lines of code in total

Reuse?

Basic widgets

```
stackIn: composite
	composite list
		title: 'Stack';
		showOnly: 10;
		display: #stack;
		format: [ :each |
			String streamContents: [:stream | each printDebugOn: stream ] ];
		with: [ :list | self installDebuggingActionsFor: list ]
```

```
stackIn: composite
	composite list
		title: 'Stack';
		showOnly: 10;
		display: #stack;
		format: [ :each |
			String streamContents: [:stream | each printDebugOn: stream ] ];
		with: [ :list | self installDebuggingActionsFor: list ]
```

Composed widgets

```
inspectorForContextAndSelfIn: composite
    composite dynamic
        display: [ :aContext |
            GTInspector new first noTitle; noActions;
                showFirst: [ :anotherComposite | anotherComposite  custom:
                    (GTDebuggerVariablesBrowser new startOn: aContext) ];
            startOn: aContext]
```

```
inspectorForContextAndSelfIn: composite
    composite dynamic
        display: [ :aContext|
            GTInspector new first noTitle; noActions;
                showFirst: [ :anotherComposite | anotherComposite  custom:
                    (GTDebuggerVariablesBrowser new startOn: aContext) ];
            startOn: aContext]
```

```
inspectorForContextAndSelfIn: composite
    composite dynamic
        display: [ :aContext|
            GTInspector new first noTitle; noActions;
                showFirst: [ :anotherComposite | anotherComposite  custom:
                    (GTDebuggerVariablesBrowser new startOn: aContext) ];
            startOn: aContext]
```

Why the debugger?

Extremely used

Extremely used in a lot of different contexts

Why change the debugger?

Contextual problems require contextual solutions

Debugging announcements

From where was this announcement trigger?

transmitter ? receiver

transmitter ? receiver

What's the connection?

next trigger

next catch

next …

Debugging program transformations

Automatically added attributes and methods

Bifrost: adds meta objects to objects

debug only the transformations

‘clean debug’

Q&A