
# Programming in the Cloud
# Context-oriented Programming for Self-supporting Development Environments

Jens Lincke and Robert Hirschfeld

Software Architecture Group
Hasso-Plattner-Institut Potsdam
www.hpi.uni-potsdam.de/swa



2012-10-24

# Outline


1. Lively Webwerkstatt:
   A Development Environment in the Cloud

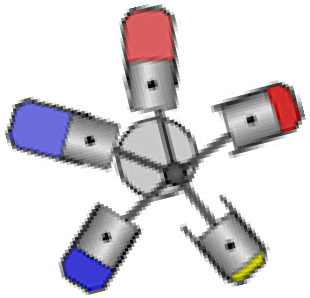
2. Lively Parts: Program Objects Directly

3. ContextJS: Evolving Self-supporting Development Environments at Run-time

# Lively Kernel


- Web-based development and runtime environment
- Lively Kernel's Promise:

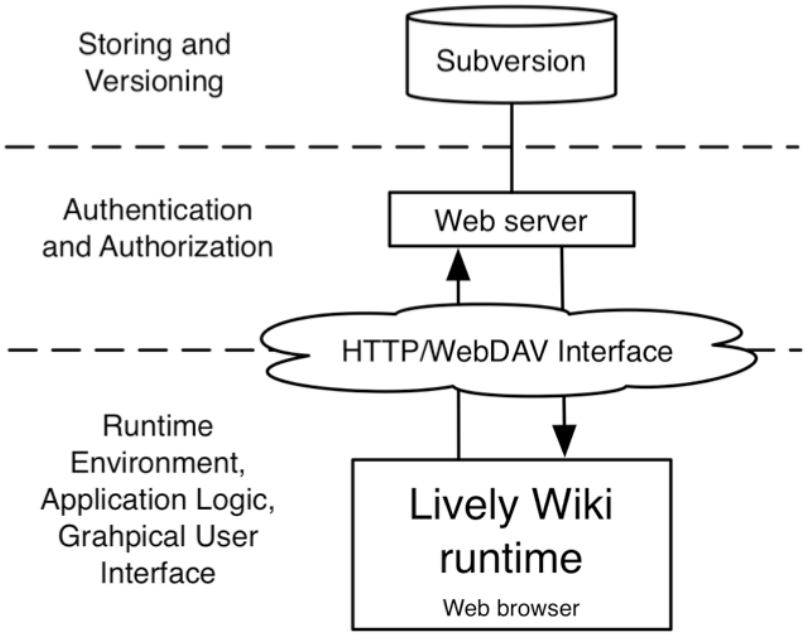
"Where ever there is Web, there is authoring"



http://lively-kernel.org/

# Lively Wiki

- Metaphor
  - "A Wiki of active objects that can be programmed by wires and tiles" [Krahn 2008]






http://lively-kernel.org/repository/webwerkstatt/demos/LivelyWikiPresentation.xhtml

# Lively Webwerkstatt

- Lively Kernel based Wiki
- Web-based Development Environment
- Core idea:
  - Allow authors to not only change content, but to shape their tools as they are using them
- Share their ideas and tools directly

→ Self-sustaining Lively Kernel Development


http://lively-kernel.org/webwerkstatt

# Parts and PartsBin


- Shared Repository of Lively Parts
- Direct object manipulation
- Deep copying of objects

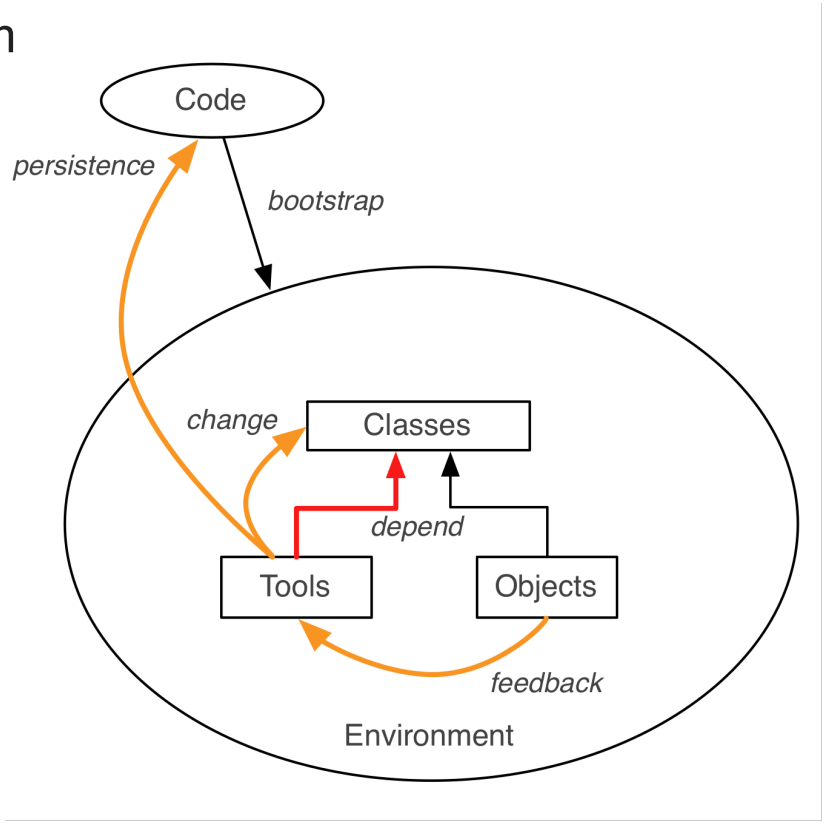
# (Meta-circular) Tools in PartsBin


- Bootstrapped to higher level development cycle
- Examples
  - Object Editor
  - PartsBin Browser
  - Inspector
  - Method Finder
- Tools are created as Parts and modified like every other item in the PartsBin

# Self-supporting Development Environments

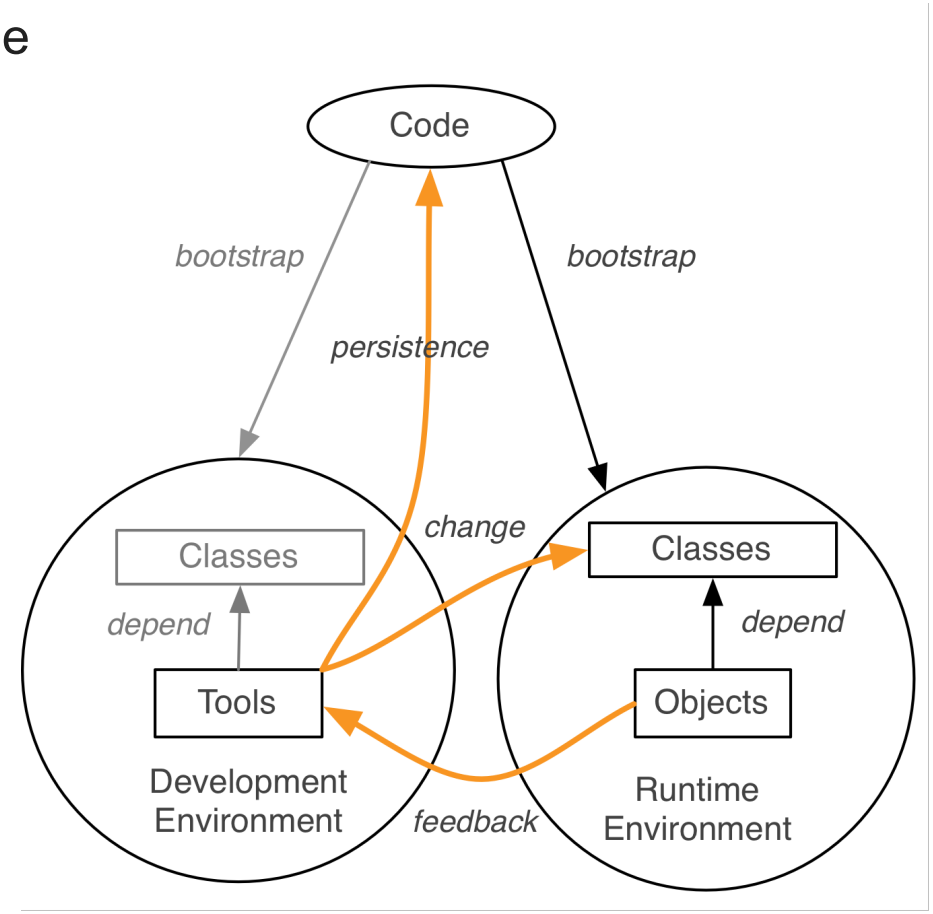

- Examples: *Smalltalk, Self, Emacs, Squeak*

- Evolve the environment while it in use
- Direct and interactive development
- But: Changes can break the system



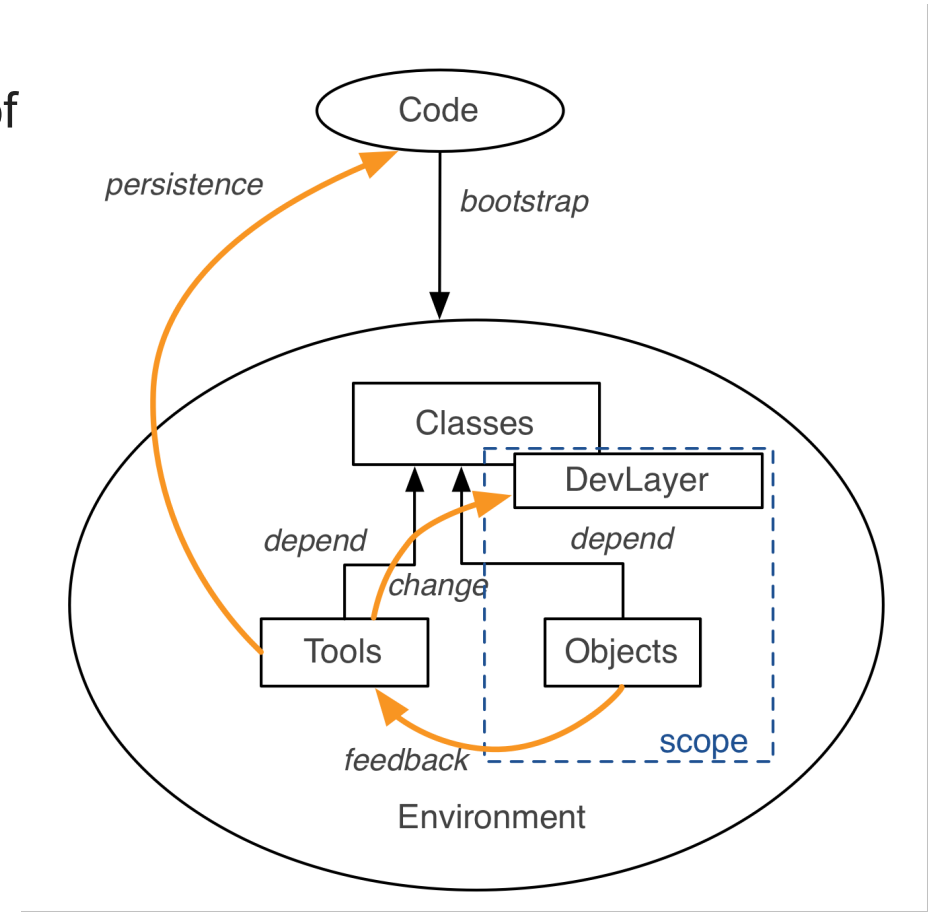
# Separate Runtime Environments

- Development tools run in a separate environment
  - Work on static code
  - Bootstrapped by external code
- Interprocess communication vs. direct access to objects

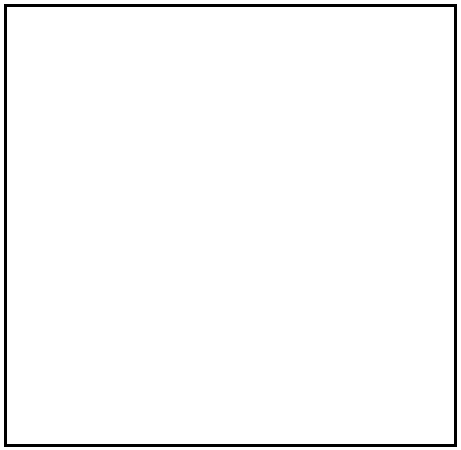


# Using Scoped Behavioral Adaptations for Evolving Self-supporting Development Environments

- Use Context-oriented Programming (COP) layers to adapt core classes and methods at run-time
- Changes affect only behavior of objects under construction



# Example 1 – Visualizing Events



```
cop.create('ShowMouseMoveLayer').refineClass(Morph, {
    onMouseMove: function(evt) {
        show(evt.mousePoint)
        return cop.proceed(evt)
    },
})
```

```
ShowMouseMoveLayer.beGlobal();
ShowMouseMoveLayer.beNotGlobal();
this.get('DebugArea').setWithLayers([ShowMouseMoveLayer])
```

# Example 2 – Text Coloring


Hello World

```
cop.create('DevLayer').refineClass(lively.morphic.Text, {
    processCommandKeys: function(evt) {
        var key = evt.getKeyChar();
        if (key) key = key.toLowerCase();
        if (evt.isShiftDown()) {  // shifted commands here...
            switch (key) {
                case "5": { this.emphasizeSelection({color: Color.black}); return true; }
                case "6": { this.emphasizeSelection({color: Color.red}); return true; }
                case "7": { this.emphasizeSelection({color: Color.green}); return true; }
                case "8": { this.emphasizeSelection({color: Color.blue}); return true; }
            }
        }
        return cop.proceed(evt);
    }
})
```

# Example 3 – Developing Autocompletion

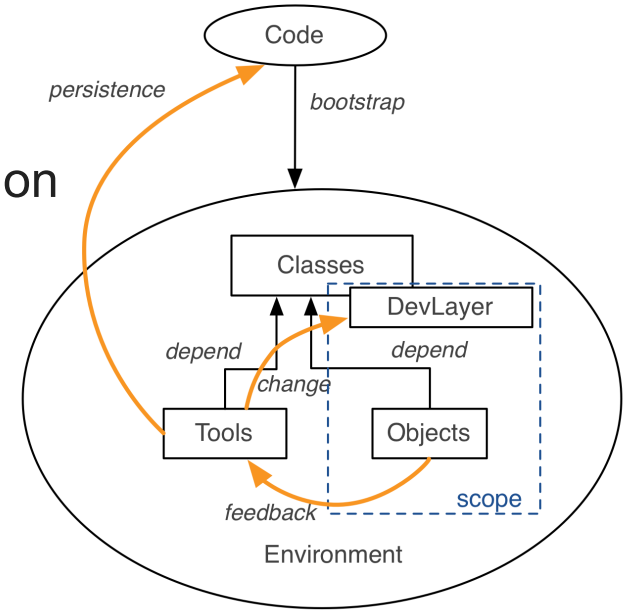

this.onMouseDown

```
cop.create('AutoCompletionLayer').refineClass(lively.morphic.Text, {
    onKeyPress: function(evt) {
        var key = evt.getKeyChar();
        if (!key.match(/\w/)) {
            this.hideWordCompletionMorph();
            return;
        }
        var range  = this.getSelectionRange()
        var cursor = range[0];

        if (cursor > 0) {
            var lastWord;

            if (lastWord = this.getLastWord()) {
                lastWord +=  key // weird errors when we proceed before our code
            }
```

# Development Layers

- Evolving tools in self-supporting development environments is direct and interactive
- Changing core parts can accidentally break the system
- We applied Context-oriented Programming to self-supporting development environments
  - Encapsulate changes into layers
  - Scope changes to objects under construction

→Work safely on new features



# Summary


1. Lively Webwerkstatt:
   A Development Environment in the Cloud

2. Lively Parts: Program Objects Directly

3. ContextJS: Evolving Self-supporting Development Environments at Runtime


# Programming in the Cloud
# Context-oriented Programming for Self-supporting Development Environments

Jens Lincke and Robert Hirschfeld

Software Architecture Group
Hasso-Plattner-Institut Potsdam
www.hpi.uni-potsdam.de/swa

2012-10-24