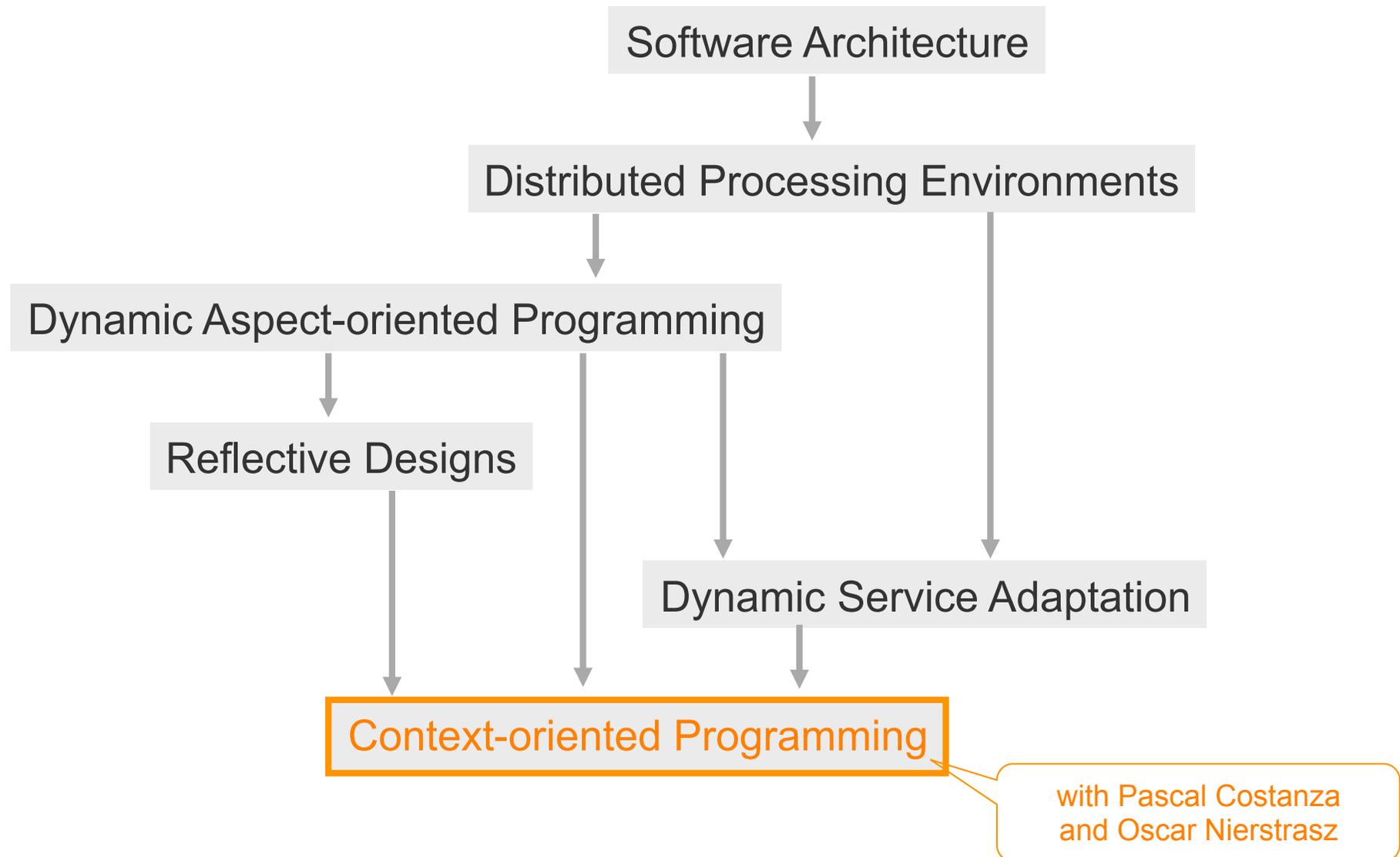


Dynamic Software Composition for Run-time System Evolution (Context-oriented Programming at HPI)

Robert Hirschfeld
Hasso Plattner Institute
University of Potsdam
Germany
<http://www.hpi.de/swa/>

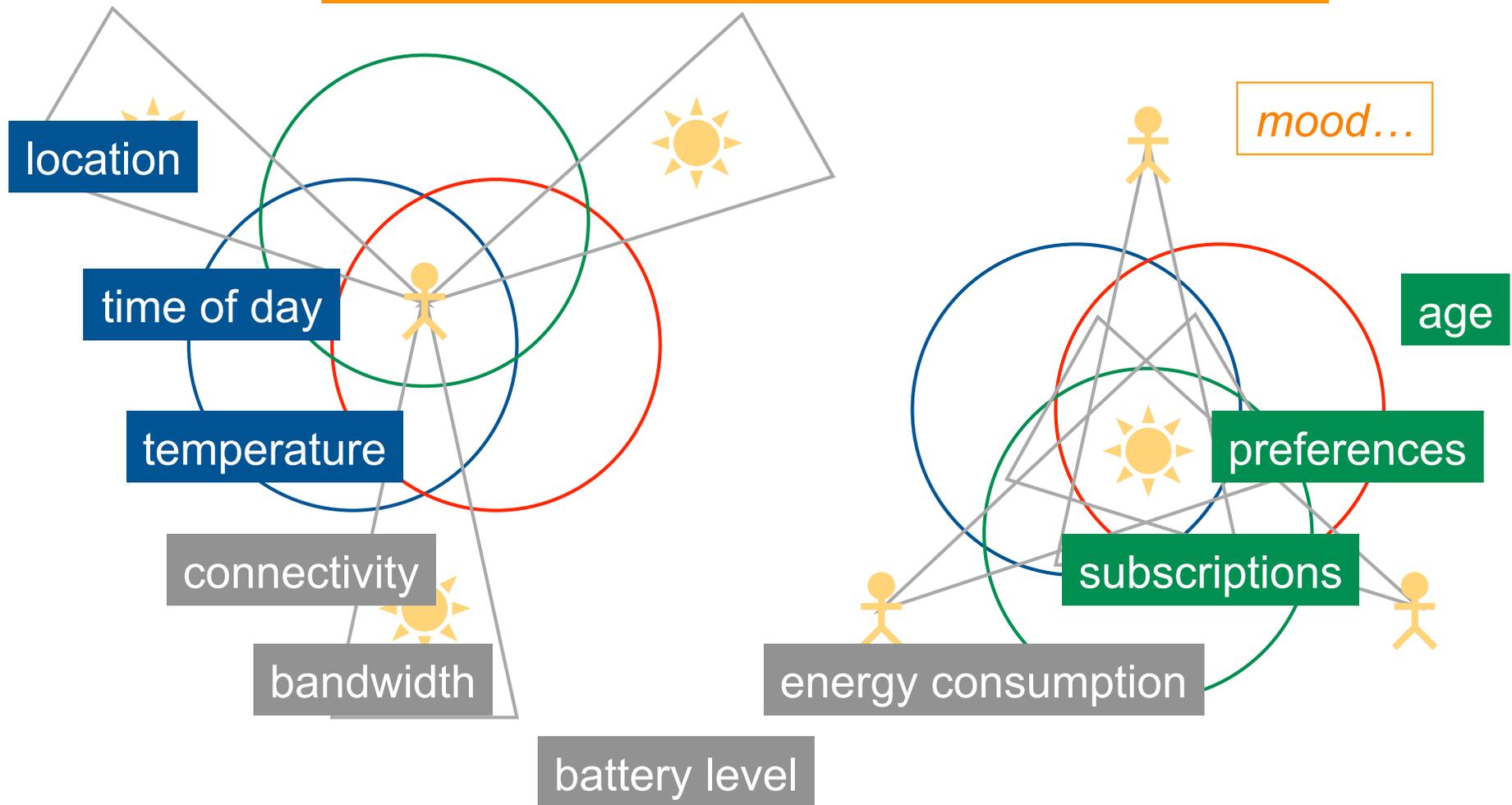
Shonan Village Center, Hayama, Kanagawa, Japan
2015-09-07

Some History...

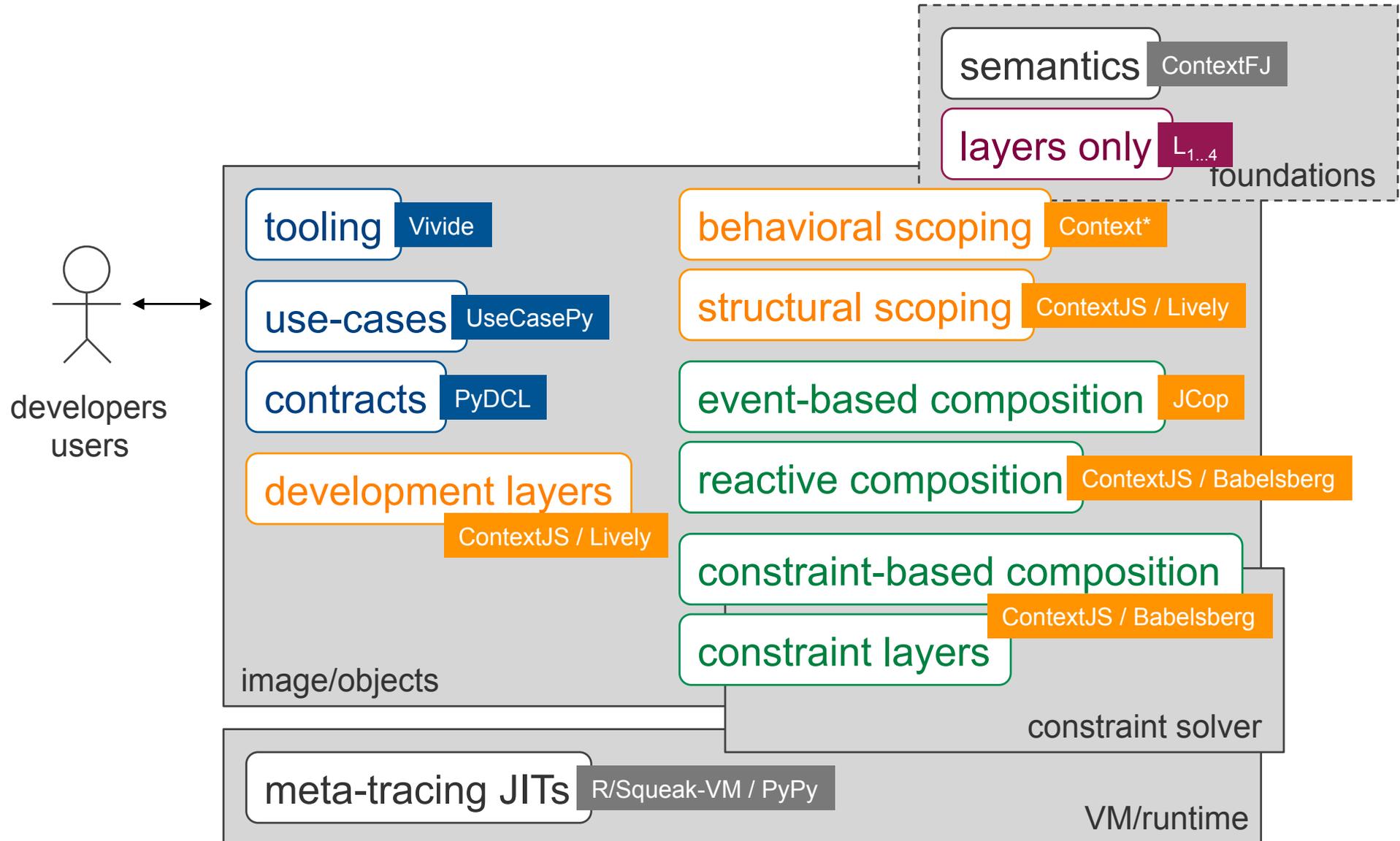


Context

context = everything computationally accessible



Outline



Behavioral Variations

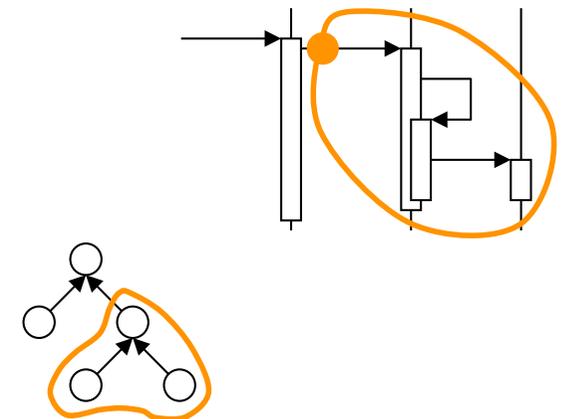
- **Behavioral** (dynamic) scoping
 - Dynamic extent of execution
 - Almost **all COP extensions**
- **Structural** (topological) scoping
 - ContextJS
 - **Development layers**
- Open implementation (OI) for scoping strategies
 - Allows for domain-specific scoping
 - Mainly applied to UI framework structures
 - Lively: Morphic
 - Webwerkstatt : Parts

behavioral scoping Context*

structural scoping ContextJS / Lively

development layers

ContextJS / Lively



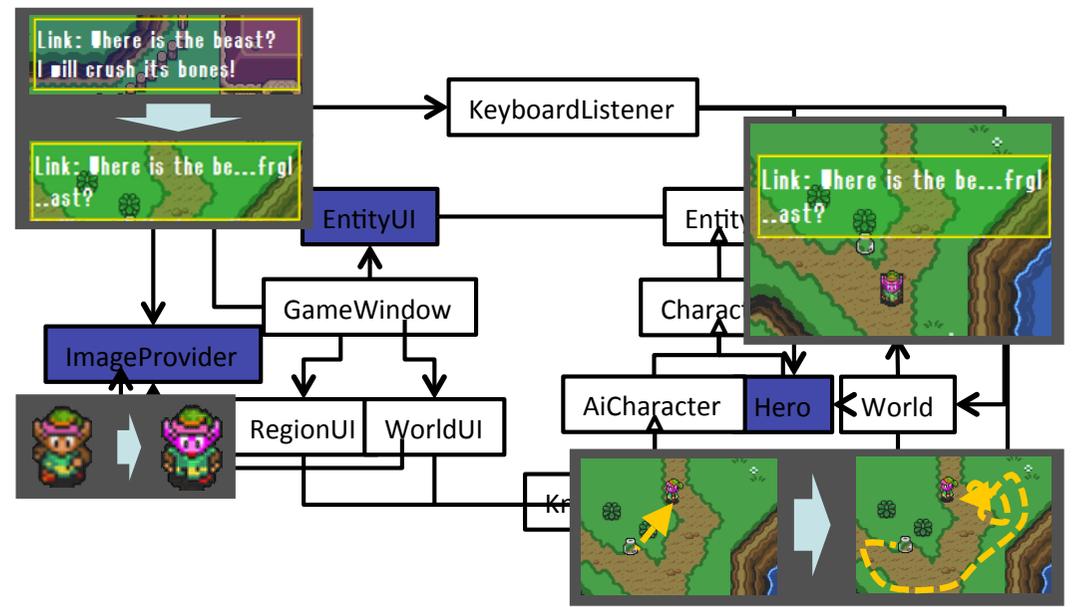
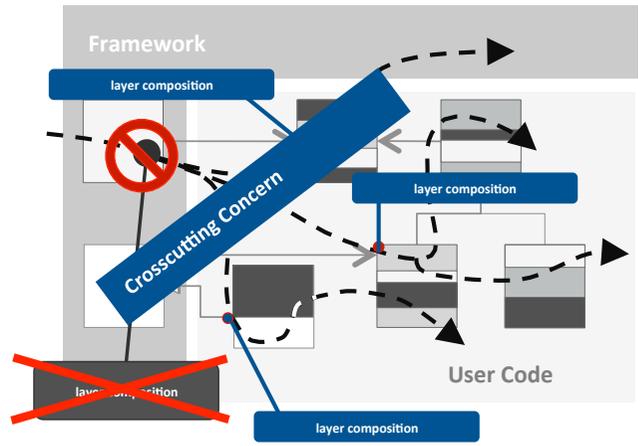
Reactive Approaches

event-based composition JCop

reactive composition ContextJS / Babelsberg

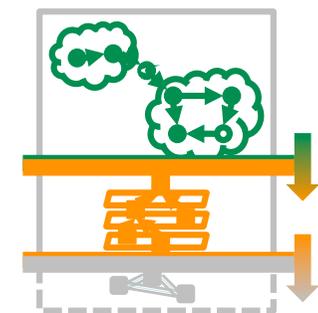
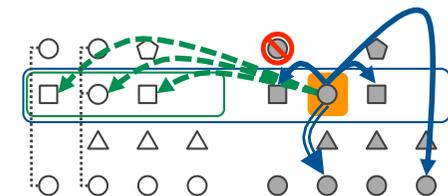
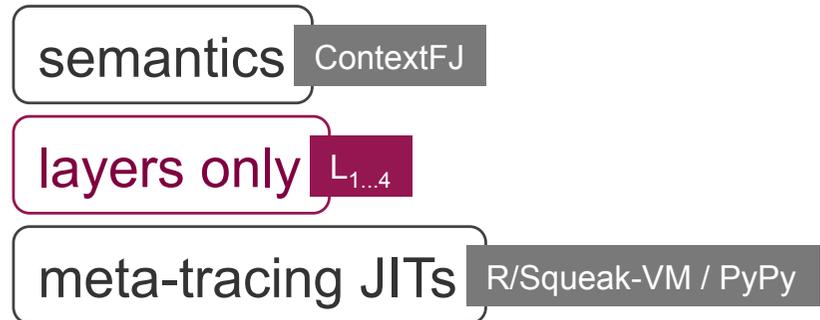
constraint-based composition ContextJS / Babelsberg

constraint layers



Foundations

- Semantics and types
 - ContextFJ
- **Symmetry**
 - No classes, only layers
 - No base system
 - $L_{1..4}$
- Sideways composition very expensive
 - **Runtime support for optimizations**
 - Meta-tracing JITs
 - R/Squeak-VM
 - Higher performance → more (meta-level) flexibility



$$\frac{PT(m, C, L_0) \text{ undefined} \quad mbody(m, C, \bar{L}', \bar{L}) = \bar{x}.e \text{ in } D, \bar{L}''}{mbody(m, C, (\bar{L}'; L_0), \bar{L}) = \bar{x}.e \text{ in } D, \bar{L}''}$$

Behavioral Scoping

COP

w/ Pascal Costanza and Oscar Nierstrasz

```
class Person {
  private String name, address;
  private Employer employer;

  Person(String newName,
         String newAddress,
         Employer newEmployer) {
    this.name = newName;
    this.employer = newEmployer;
    this.address = newAddress;
  }

  String toString() {return "Name: "+name;}
```

```
layer Address {
  String toString() {
    return proceed()+"; Contact: "+address;
  }
}
```

```
layer Employment {
  String toString() {
    return proceed()+"; [Employer] "+employer;
  }
}
```

```
class Employer {
  private String name, address;

  Employer(String newName,
           String newAddress) {
    this.name = newName;
    this.employer = newEmployer;
  }

  String toString() {return "Name: "+name;}
```

```
layer Address {
  String toString() {
    return proceed()+"; Visitors: "+address;
  }
}
```

```
class Person {
```

```
private Employer hpi = new Employer("HPI", "14440 Potsdam");
private Person robert = new Person("Robert Hirschfeld", "14471 Potsdam", hpi);
```

```
Person
```

```
System.out.println(robert);
```

```
this
```

```
this
```

```
this
```

```
}
```

Output: Name: Robert Hirschfeld

```
String toString() {return "Name: "+name;}
```

```
String toString() {return "Name: "+name;}
```

```
layer Address {
  String toString() {
    return proceed()+"; Contact: "+address;
  }
}
```

```
layer Address {
  String toString() {
    return proceed()+"; Visitors: "+address;
  }
}
```

```
layer Employment {
  String toString() {
    return proceed()+"; [Employer] "+employer;
  }
}
```

```
class Person {
```

```
private Employer hpi = new Employer("HPI", "14440 Potsdam");
private Person robert = new Person("Robert Hirschfeld", "14471 Potsdam", hpi);
```

```
Person
```

```
    with (Address) {
        System.out.println(robert);
    }
    this
    this
    this
}
```

Output: Name: Robert Hirschfeld; Contact: 14471 Potsdam

```
String toString() {return "Name: "+name;}
```

```
String toString() {return "Name: "+name;}
```

```
layer Address {
    String toString() {
        return proceed()+"; Contact: "+address;
    }
}
```

```
layer Address {
    String toString() {
        return proceed()+"; Visitors: "+address;
    }
}
```

```
layer Employment {
    String toString() {
        return proceed()+"; [Employer] "+employer;
    }
}
```

```
class Person {
```

```
    private Employer hpi = new Employer("HPI", "14440 Potsdam");
    private Person robert = new Person("Robert Hirschfeld", "14471 Potsdam", hpi);
```

```
    Person with (Employment) {
        with (Address) {
            System.out.println(robert);
        }
    }
    this }
    this }
    this }
}
```

Output: *Name: Robert Hirschfeld; Contact:14471 Potsdam; [Employer] Name: HPI; Visitors: 14440 Potsdam*

```
String toString() {return "Name: "+name;}
```

```
String toString() {return "Name: "+name;}
```

```
layer Address {
    String toString() {
        return proceed()+"; Contact: "+address;
    }
}
```

```
layer Address {
    String toString() {
        return proceed()+"; Visitors: "+address;
    }
}
```

```
layer Employment {
    String toString() {
        return proceed()+"; [Employer] "+employer;
    }
}
```

```
class Person {
```

```
    private Employer hpi = new Employer("HPI", "14440 Potsdam");
    private Person robert = new Person("Robert Hirschfeld", "14471 Potsdam", hpi);
```

```
    Person with (Address) {
        with (Employment) {
            System.out.println(robert);
        }
    }
    this }
    this }
    this }
}
```

Output: Name: Robert Hirschfeld; [Employer] Name: HPI;
Visitors: 14440 Potsdam; Contact: 14471 Potsdam

```
String toString() {return "Name: "+name;}
```

```
String toString() {return "Name: "+name;}
```

```
layer Address {
    String toString() {
        return proceed()+"; Contact: "+address;
    }
}
```

```
layer Address {
    String toString() {
        return proceed()+"; Visitors: "+address;
    }
}
```

```
layer Employment {
    String toString() {
        return proceed()+"; [Employer] "+employer;
    }
}
```

```
class Person {
```

```
private Employer hpi = new Employer("HPI", "14440 Potsdam");
private Person robert = new Person("Robert Hirschfeld", "14471 Potsdam", hpi);
```

```
Person with (Address) {
    with (Employment) {
        System.out.println(robert);
    }
}
this
this
this
```

Thread-1

Output-1: Name: Robert Hirschfeld; [Employer] Name: HPI;
Visitors: 14440 Potsdam; Contact: 14471 Potsdam

```
String with (Employment) {
    with (Address) {
        System.out.println(robert);
    }
}
layer
String
re
```

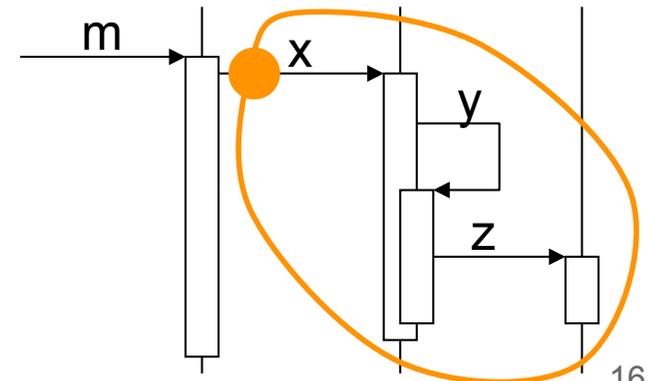
Thread-2

Output-2: Name: Robert Hirschfeld; Contact: 14471 Potsdam;
[Employer] Name: HPI; Visitors: 14440 Potsdam

```
layer
String
return proceed()+" "; [Employer] "+employee;
```

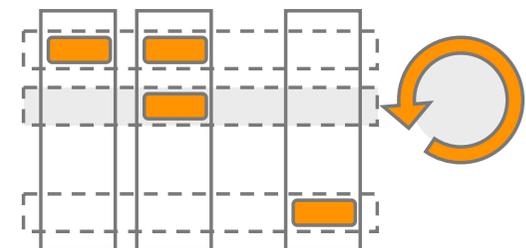
Dynamically-scoped Layer Activation

- Constructs
 - `with (...)` {...}
 - `without (...)` {...}
 - `next (...)`
- Activate (deactivate) layers for the **current thread**
 - **No interference** with other layer activations/deactivations in other threads
 - Layers are activated/deactivated only for the **dynamic extent** of the associated code block (dynamic scoping)
 - **Activation order** determines method precedence



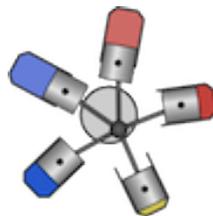
COP Basics Summary

- **Behavioral variations**
 - Partial class, method, and layer definitions
- **Layers**
 - Groups of related context-dependent behavioral variations
- **Activation**
 - Activation and deactivation of layers at run-time
- **Context**
 - Anything computationally accessible
- **Scoping**
 - Well-defined explicitly-controlled scopes



COP Extensions (Some...)

- ContextS
- ContextS2
- ContextJS
- JCop (ContextJ)
- ContextPy
- PyDCL
- UseCasePy
- PyContext
- ContextR
- ContextG
- ContextAmber
- $L_{1...4}$



- ContextL
- ContextScheme
- ContextJ*
- ContextErlang
- EventCJ
- Lambic
- Ambience
- COP.JS
- deIMDSCO/cj
- Phenomenal Gem
- Subjective-C
- Context Petri Nets



Structural Scoping & Development Layers

Strect

w/ Jens Lincke

Lively Webwerkstatt

HeatingSystem

Heat: 6.4681 [kW/s]
 Stopped: false
 Factor: 20
 Max: 5 [kW]

```

if ($heat < full && #fuel.$gas > $.Max * at &&
    #HeatStorage.$Energy < 0.3 * #HeatStorage.$Max) {
    var gas = $.Max
    #Fuel.$Gas -- g
    $.Heat += gas
}
        
```

\$.Factor
\$.Heat
\$.Max
\$.Stopped

SerializationInspector

min refs: 0 min full size: 20000

Object	Name	ClassName
0	root	lively.morphic
1	submorphs	object
2	146	PartsBinBrowser
3	attributeConnections	lively.morphic
4	0	object
5	0	AttributeConne
6	0	lively.morphic
7	0	
8	0	
9	0	
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	
16	0	
17	3	
18	0	
19	0	
20	0	
21	0	
22	1	shape
23	1	stringifiedShapeNode
24	0	
25	0	
26	0	
27	0	
28	0	
29	2	shape
30	2	stringifiedShapeNode

URL: http://lively-kernel.org/repository/webwerkstatt/users/jenslincke/thesis/data/svn_log_jenslincke.xml

Filter: *\$ Min: 1

maximize SpreadsheetScripter.metainfo, markoeroder, Mon May 09 2011 21:35:07 GMT+0200 (CEST)

LaTeX Outline

compare print

- Examples? [03_sdlis]
- but... [03_sdlisupport1]
- LT: Maybe make the
- REVIEWER 2: "In t
- [04_malleable_tools.tex:49]
- but... [04_malleable_tools.tex:62]
- add figure of StyleEditor [04_malleable_tools.tex:99] @done (13-12-04 14:10)
- add figure of dragging object with halo? [04_malleable_tools.tex:100] @done (13-12-04 14:10)
- Problems with serializing scripts? [04_malleable_tools.tex:106]
- Active Content -> Application Content -> (just) Content [04_malleable_tools.tex:114]
- make clear how we can live with changes [04_malleable_tools.tex:183]
- Should I present the Syntax and Semantics of ContextJS in a more abstract way? [05_devlayers]
- How did Maite present the Syntax of ContextJ? [05_devlayers.tex:51]
- write section "Adapting Core Behavior from Scripted Tools" [05_devlayers.tex:286]
- implement section "Repository of Parts" [07_partsbin.tex:8]
- insert ref to code duplication [07_partsbin.tex:50]
- only random number based UUIDs can be generated in the browser. [07_partsbin.tex:57]
- write section "Scripting Parts - Summary" [07_partsbin.tex:80]
- write section "Case Study: Server Side Scripting" [08_casestudies.tex:58]
- Ref? [08_casestudies.tex:334]
- show tracing result [08_casestudies.tex:472]
- to be fair, we could write down non COP tracer code that manages dynamic scope somehow
- case-study chapter: write summary for [08_casestudies.tex:739]
- Reviewer 1: 'What is the purpose of this "related work" section except mentioning some relat

World

name: 10 Rectangle [lively.morphic.Shapes.Rectangle]

ignore -> <- do not ignore

__SourceModuleName__: Global.lively.morphic....

__Position__: [object Object]

__Extent__: [object Object]

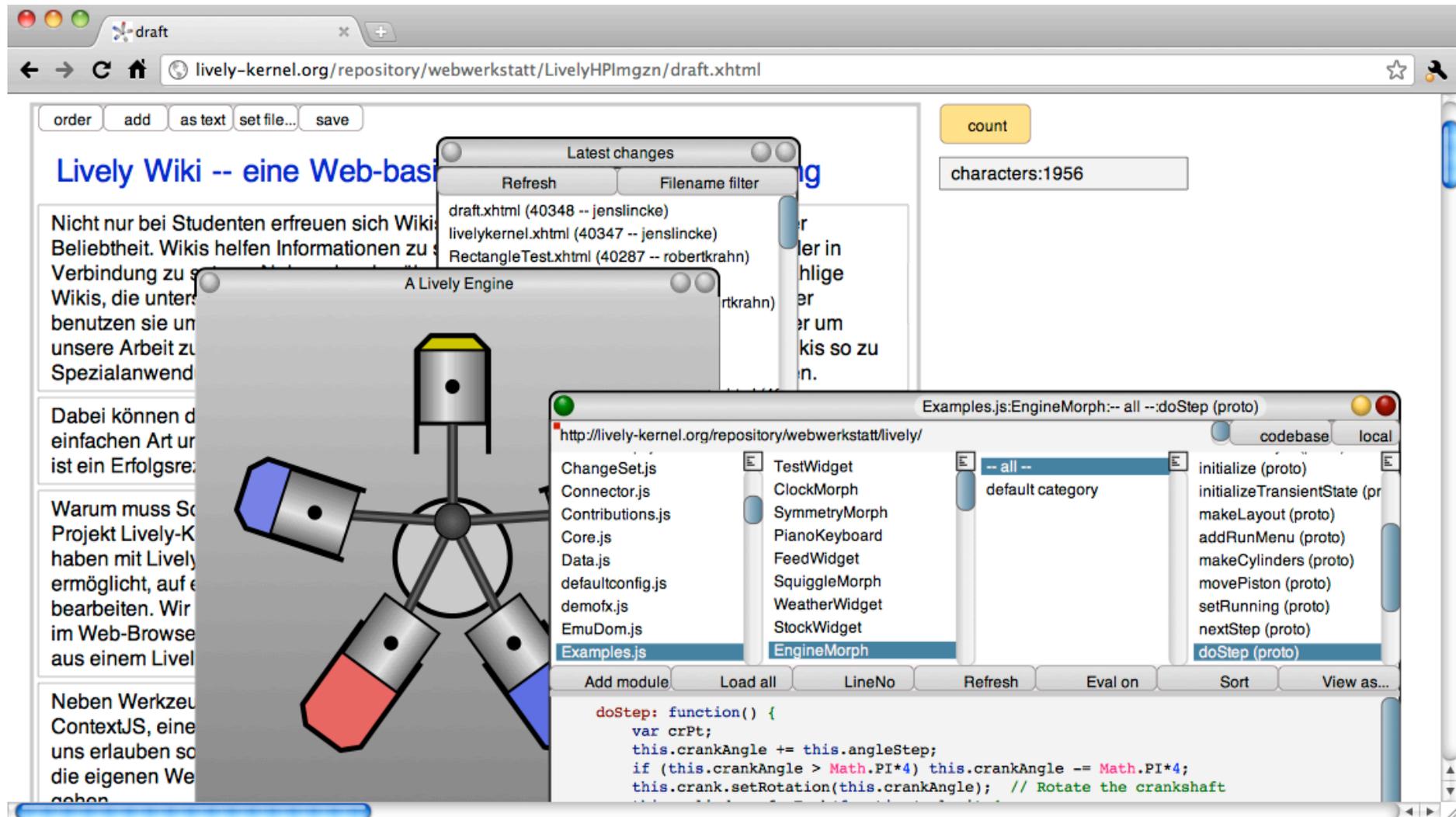
__Fill__: [object Object]

renderContextTable: [object Object]

__LivelyClassName__: lively.morphic.Shapes....

7 Point
4 undefined
2 lively.morphic.EventHandler
2 Color
2 lively.morphic.Shapes.Recte
1 ChangeSet
1 lively.morphic.HandMorph
1 lively.morphic.World

Lively Kernel and Lively Wiki



The screenshot shows a web browser window with the URL `lively-kernel.org/repository/webwerkstatt/LivelyHPIimgzn/draft.xhtml`. The page content includes a title "Lively Wiki -- eine Web-basierte Wiki" and several paragraphs of German text. A diagram of a crankshaft is visible in the center. Overlaid on the page are several windows:

- Latest changes**: A window showing a list of recent changes:
 - draft.xhtml (40348 -- jenslincke)
 - livelykernel.xhtml (40347 -- jenslincke)
 - RectangleTest.xhtml (40287 -- robertkrahn)
- Examples.js:EngineMorph:-- all --:doStep (proto)**: A code editor window showing the following code:


```
doStep: function() {
  var crPt;
  this.crankAngle += this.angleStep;
  if (this.crankAngle > Math.PI*4) this.crankAngle -= Math.PI*4;
  this.crank.setRotation(this.crankAngle); // Rotate the crankshaft
```

Self-supporting Development Environments

- Collaboratively evolve tools and environment
 - Adapt tools while using them
 - From within
 - Share easily
- Design goals for self-supporting development environments (SSDEs)
 - Innovative repair
 - Short feedback loops → immediacy
- Technical problem
 - Changes to core functionality might break the environment (also for everyone)

ContextJS

- Library-based COP extension to JavaScript
- Open implementation (OI) for layer composition
 - Behavioral scoping
 - **Structural scoping**

```

EventCounter = {
  n: 0,
  count: function(evt) {
    this.n = this.n + 1;
  }
}

```

```

EventCounter.count = function(evt) {
  alert("evt: " + evt);
  this.n = this.n + 1;
}

```

```

cop.create("DevLayer").refineObject(EventCounter, {
  count: function(evt) {
    alert("evt: " + evt);
    this.n = this.n + 1;
  }
})

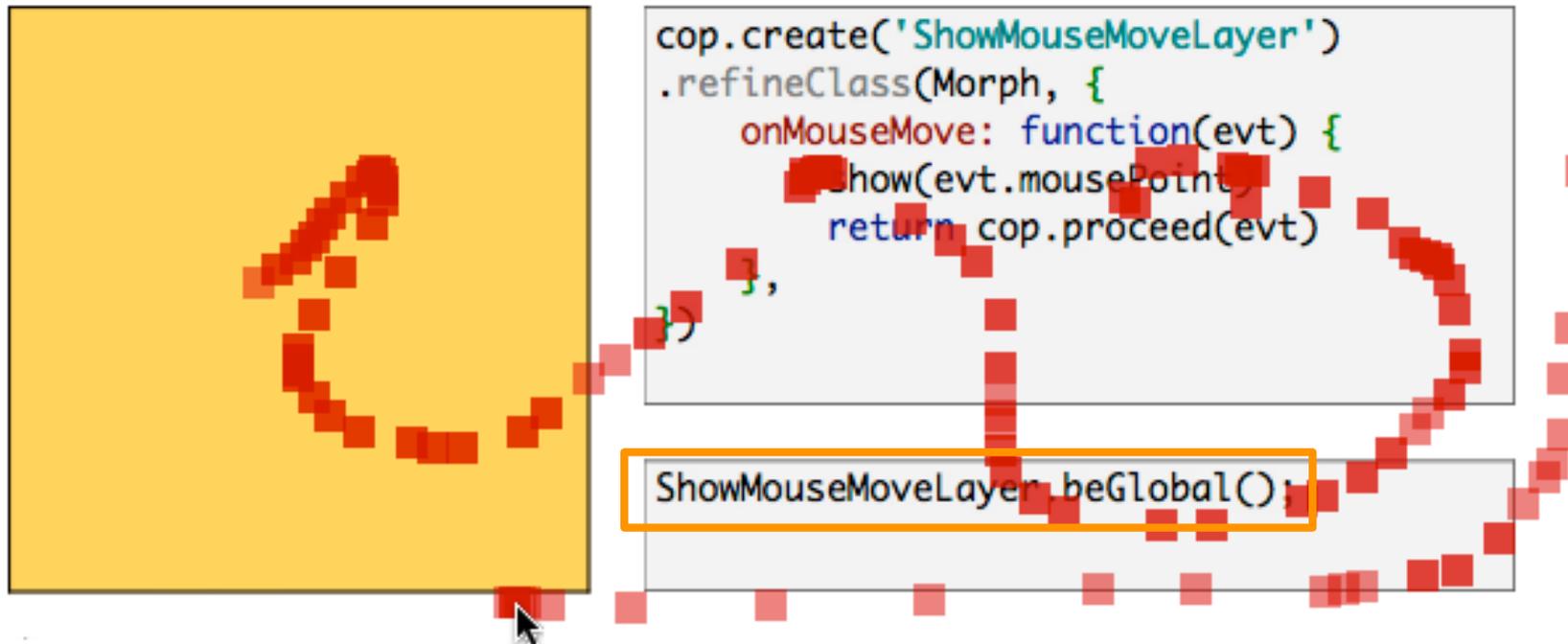
```

```
cop.proceed(evt);
```

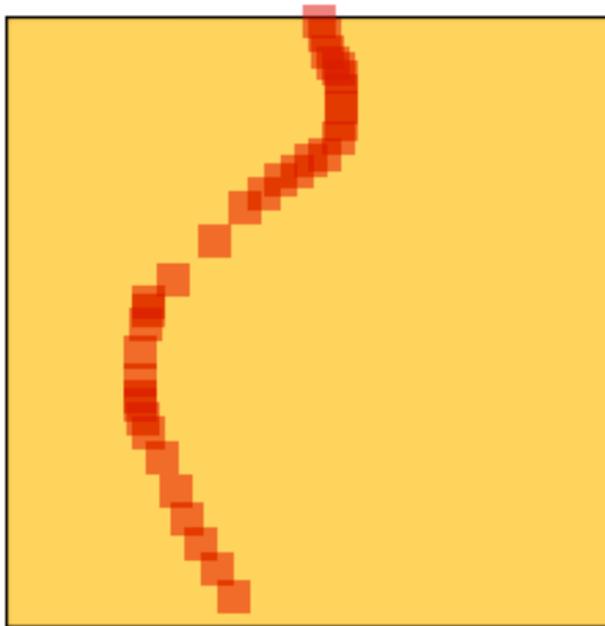
```
DevLayer.beGlobal();
```

```
debugArea.setWithLayers([DevLayer])
```

Example 1: Visualizing Events



Example 1: Visualizing Events



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

```
$morph('DebugArea').setWithLayers([ShowMouseMoveLayer])
```

Example 2: Text Coloring

Hello World

```
M Workspace
cop.create('TextColorLayer').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);
  }
});
```

```
this.setWithLayers([...])
```

```
TextColorLayer.beGlobal()
```

Example 3: Developing Auto-completion

this.onMouse

- onMouse
- onMouseDown
- onMouseDownEr
- onMouseMove
- onMouseMoveEr
- onMouseOut

DevArea

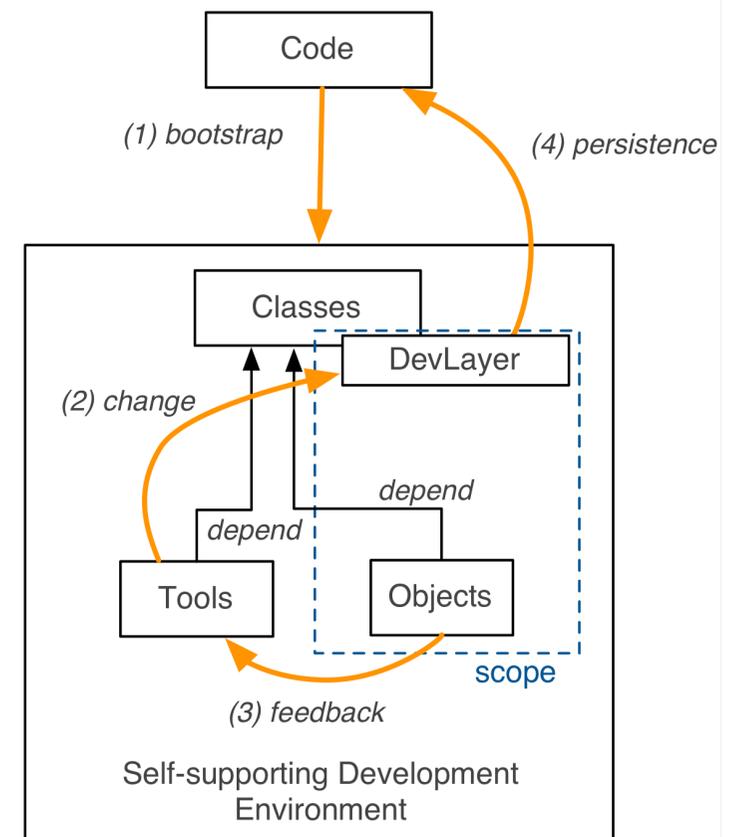
```
cop.create('WordCompletionLayer').refineClass(lively.morphic.Text, {
  onKeyPress: function(evt) {
    var key = evt.getKeyChar();
    if (!key.match(/\w/)) {
      this.hideWordCompletionMorph();
      return;
    }
    var range = this.getSelectionRange()
    var cursor = range[0];
    // ....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

Workspace

```
$morph('DevArea').setWithLayers([WordCompletionLayer]);
```

Structural Scoping Summary

- Application of **COP to SSDEs**
 - Organize changes into layers
 - Apply changes during development to only objects of interest
 - Structural scoping
 - Development layers
- **Evolution of tools** in SSDEs can be direct, interactive, and safe
- Future work
 - Refactoring of layers back into base



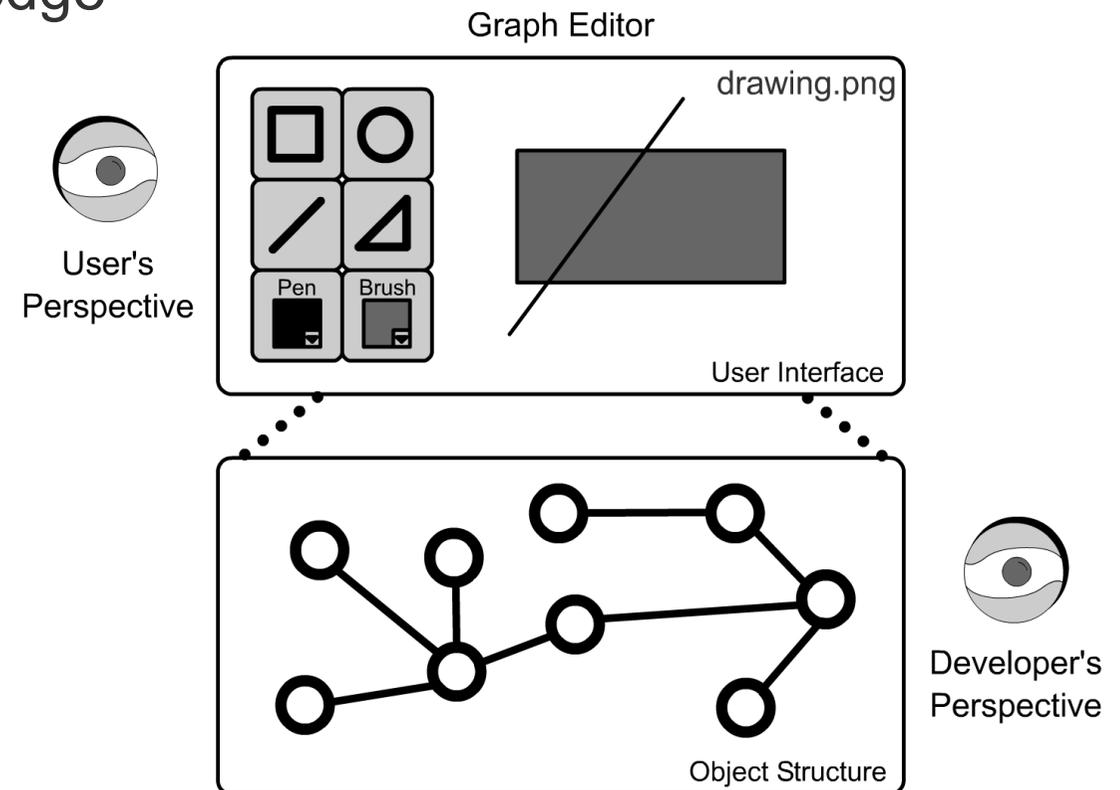
Explicit Use-case Representation

UUC

w/ Michael Perscheid

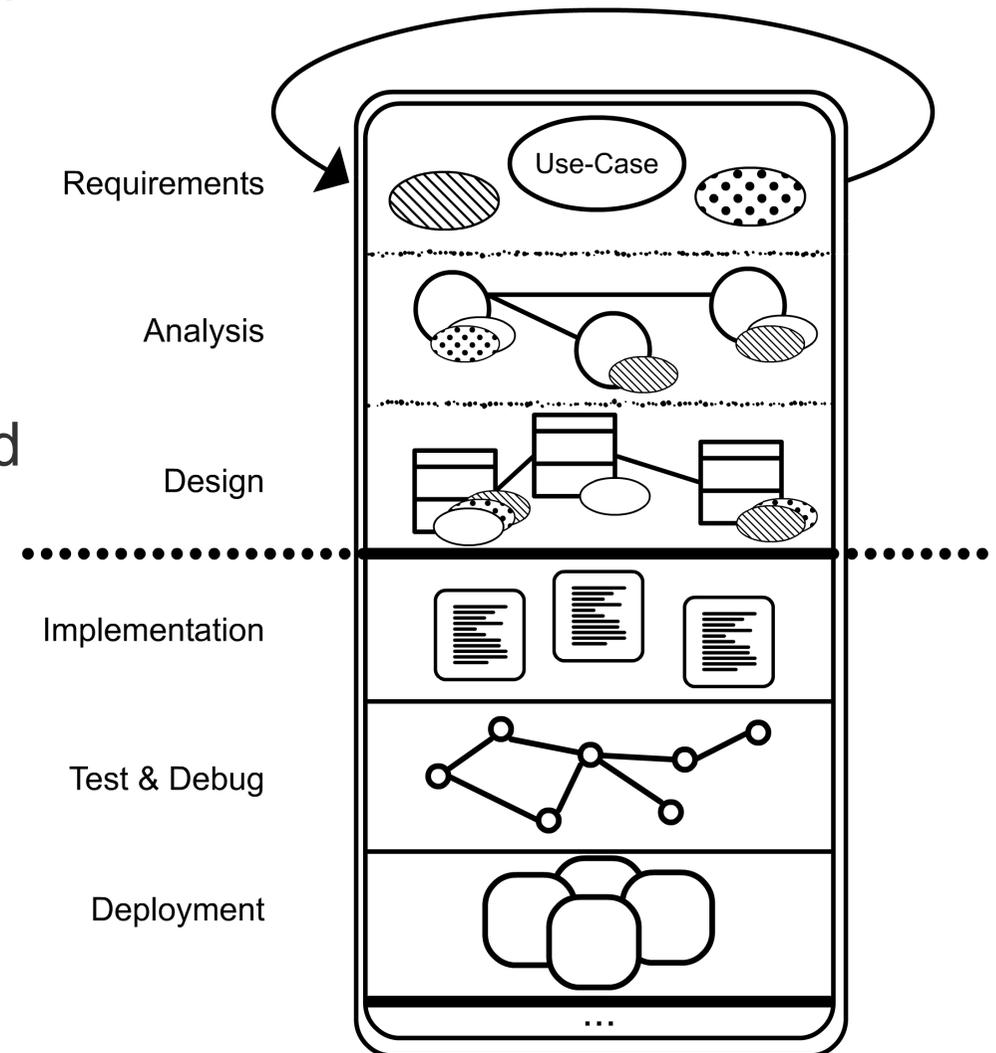
Use-cases in Software Development

- **Users** perceive program behavior without implementation knowledge
- **Developers** also know internals and implementation details
- Use-cases describe interaction at system **boundary**
- Use-cases **link** both perspectives



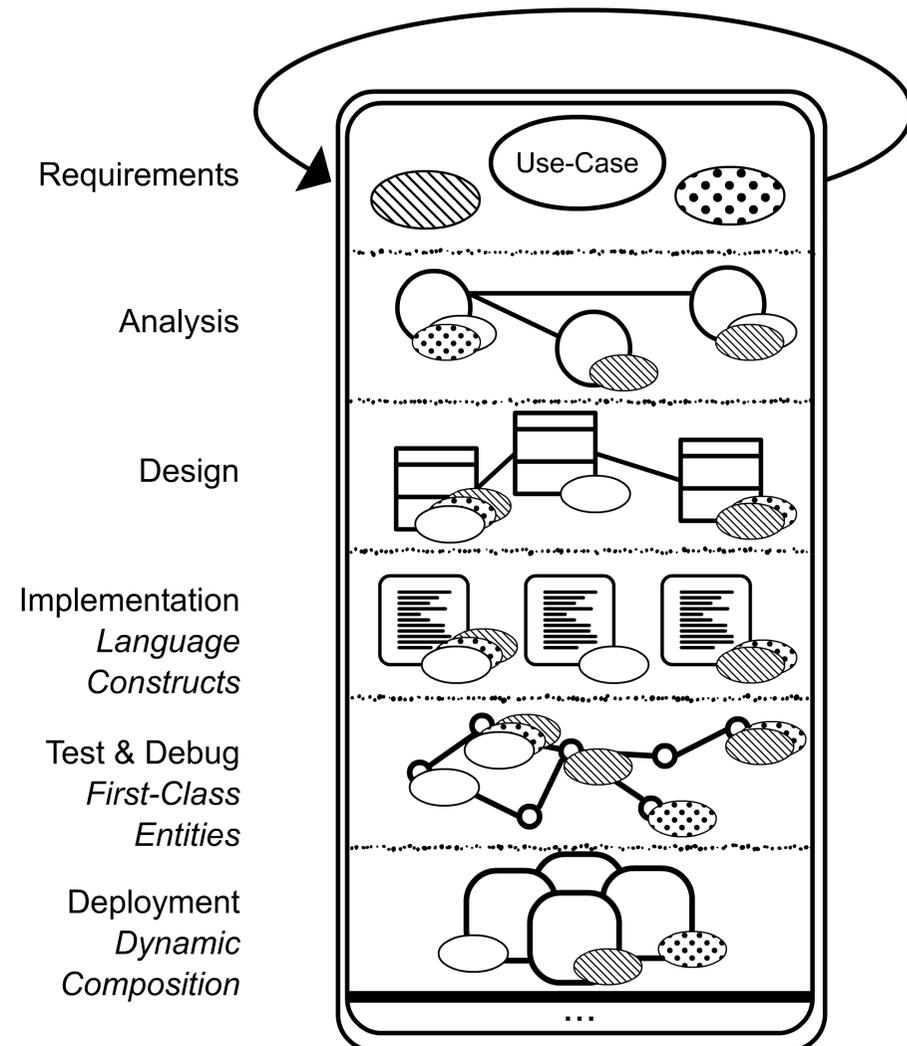
Use-cases in Software Development

- Use-cases and variants are **integral part** of most contemporary development processes
- **Traceability** to use-cases **lost** in later more code- and deployment-centric development activities
- → **Lack of understanding** about which parts of the system contribute to which use-case



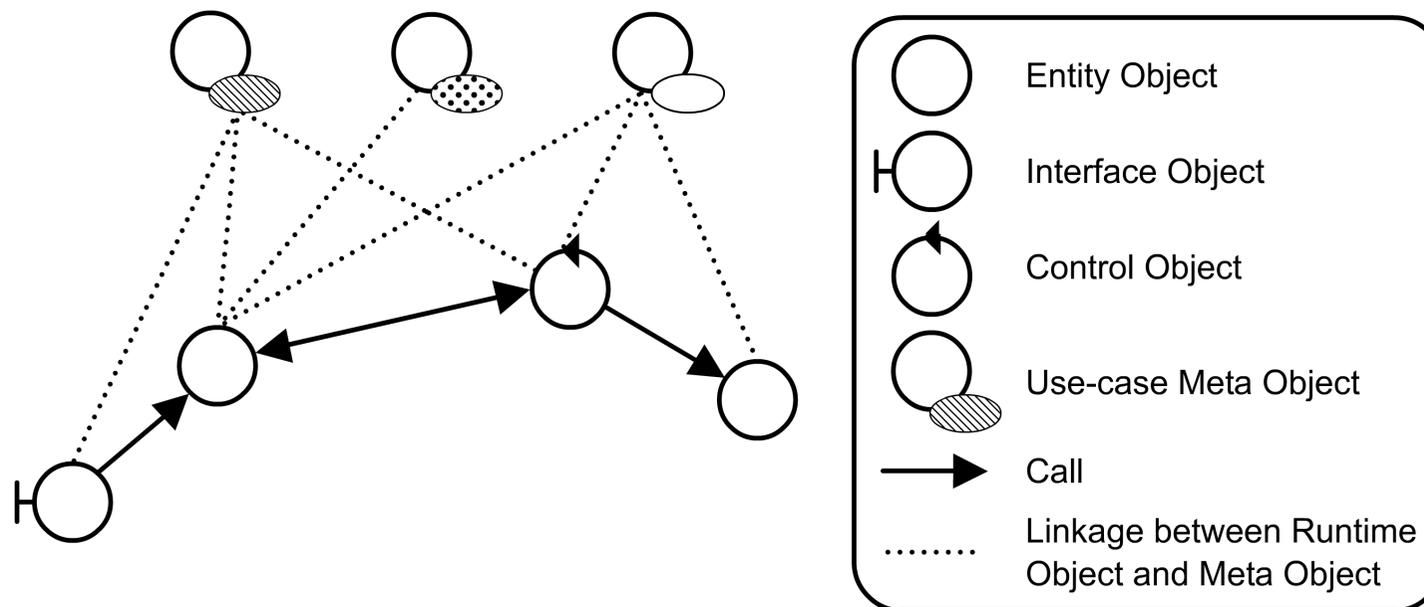
Use-case-centered Development

- Explicit use-case representation in object-oriented languages
- Use-cases in source code, as deployment units, and at run-time



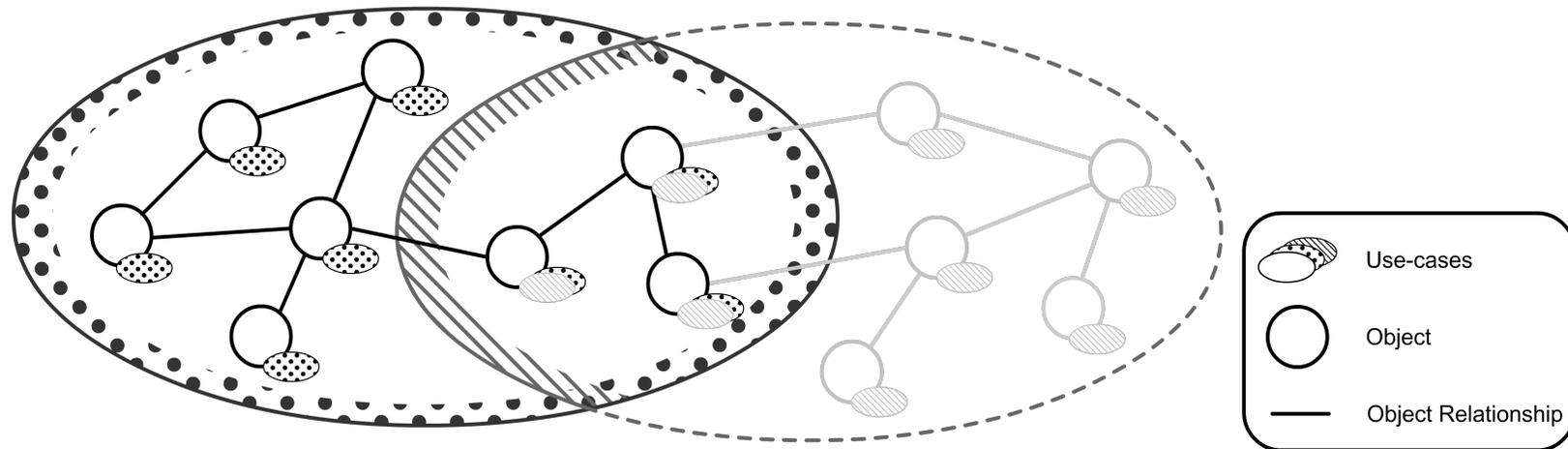
First-class Entities at Run-time

- Based on source code annotations
- Use-cases as **meta objects**
- Central registry of use-case descriptions
- **Available at run-time** for introspection and intercession



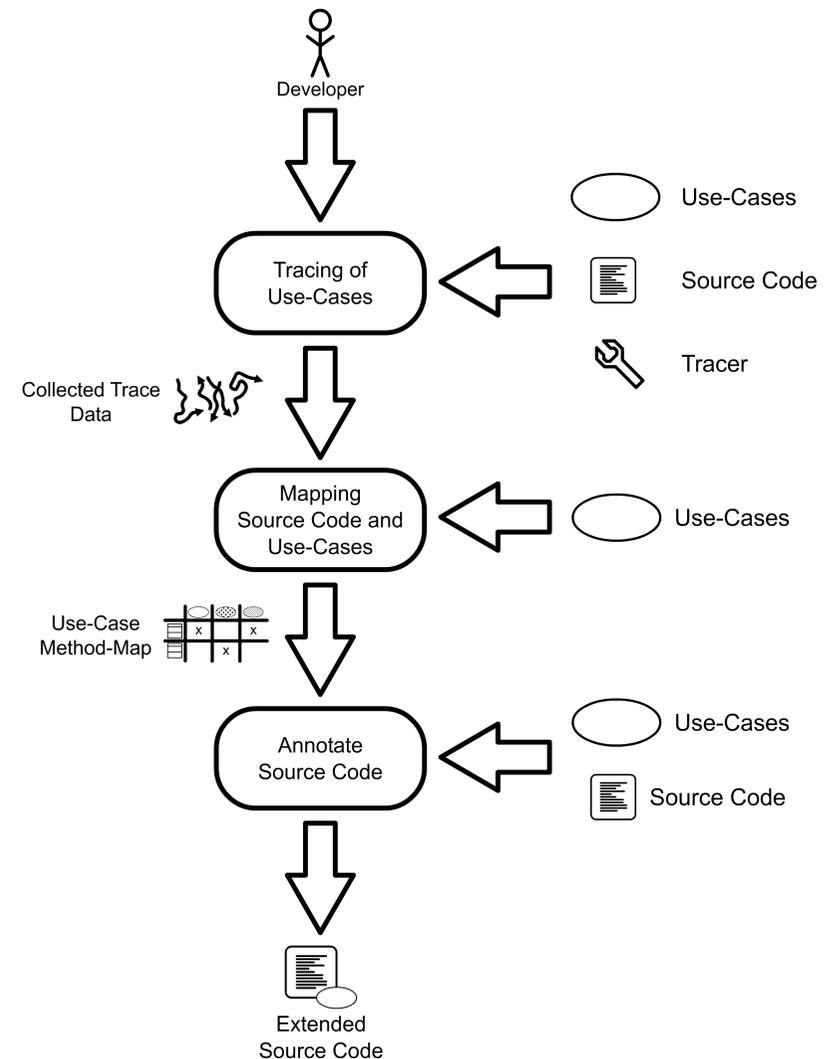
Dynamic Composition

- Based on selection of a set of desired use-cases
- Requires use-case-aware method dispatch
- Allows for use-cases as **deployment units**



Use-case Discovery

- Introduce use-case-centered development to **existing systems**
- Based on **feature location** techniques
- Tracer observes execution of use-cases from the users' point of view
- Semi-automatic and automatic implementations



Use-case Layers Summary

- Use-case-centered development allows for **explicit representation** of use-cases in code and at run-time
 - Available during implementation, testing, and deployment
 - **Use-case discovery** migrates existing systems to use-case-centered development
- Future work
 - User studies
 - Improved analysis techniques
 - Better tool support



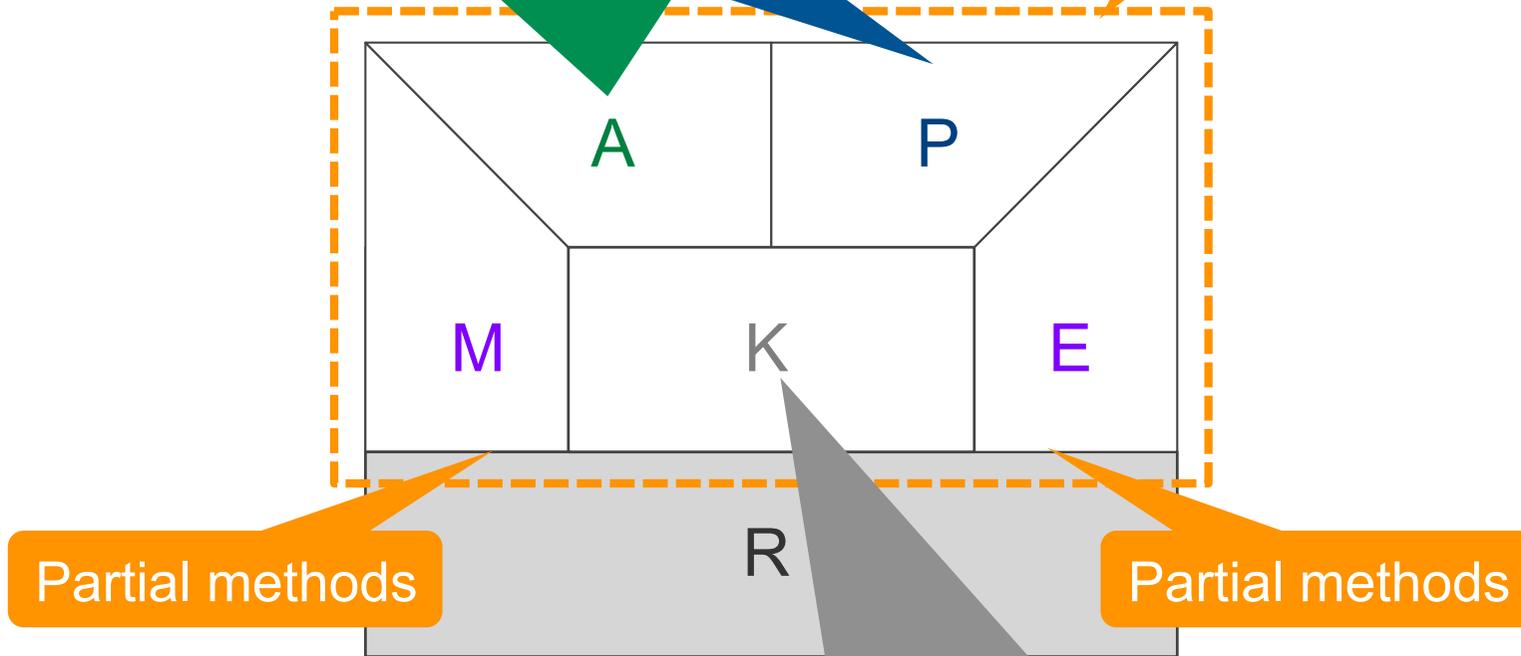
Monitor Analyze Plan Execute-Knowledge

MAPE-K

COP & MAPE-K

Regular (or partial) methods and objects

Layers



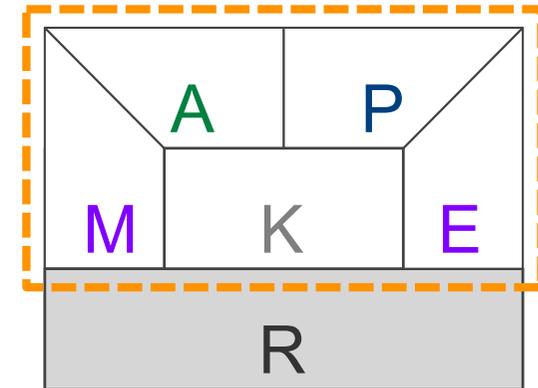
Partial methods

Partial methods

Regular (or partial) methods and objects

A first sketch...

COP & MAPE-K



```

class ResourceManager {
  // ...
  layer MAPE { // not necessary
    static mapeBefore(resource, in, ...) {
      // note = ...;
      monitorBefore(...);
      analyzeBefore(...);
      planBefore(...);
      executeBefore(...);
      return note;
    }
    static mapeAfter(resource, in, note, out, ...) {
      // newOut = ...
      monitorAfter(...); // executeAfter(...);
      analyzeAfter(...); // planAfter(...);
      planAfter(...); // analyzeAfter(...);
      executeAfter(...); // monitorAfter(...);
      return newOut;
    }
  }
}

```

```

class ManagedResource {
  // ...
  process(in) {
    // out := ...;
    return out;
  }
  layer MAPE {
    process(in) {
      note := ResourceManager.mapeBefore(self, in, ...);
      out := next(in);
      out := ResourceManager.mapeAfter(self, in, note, out, ...);
      return out;
    }
  }
}

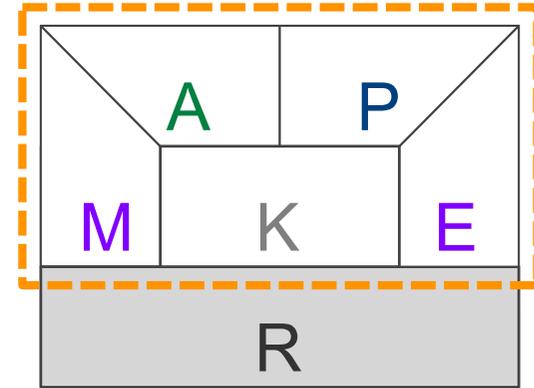
```

A first sketch...

COP & MAPE-K

```

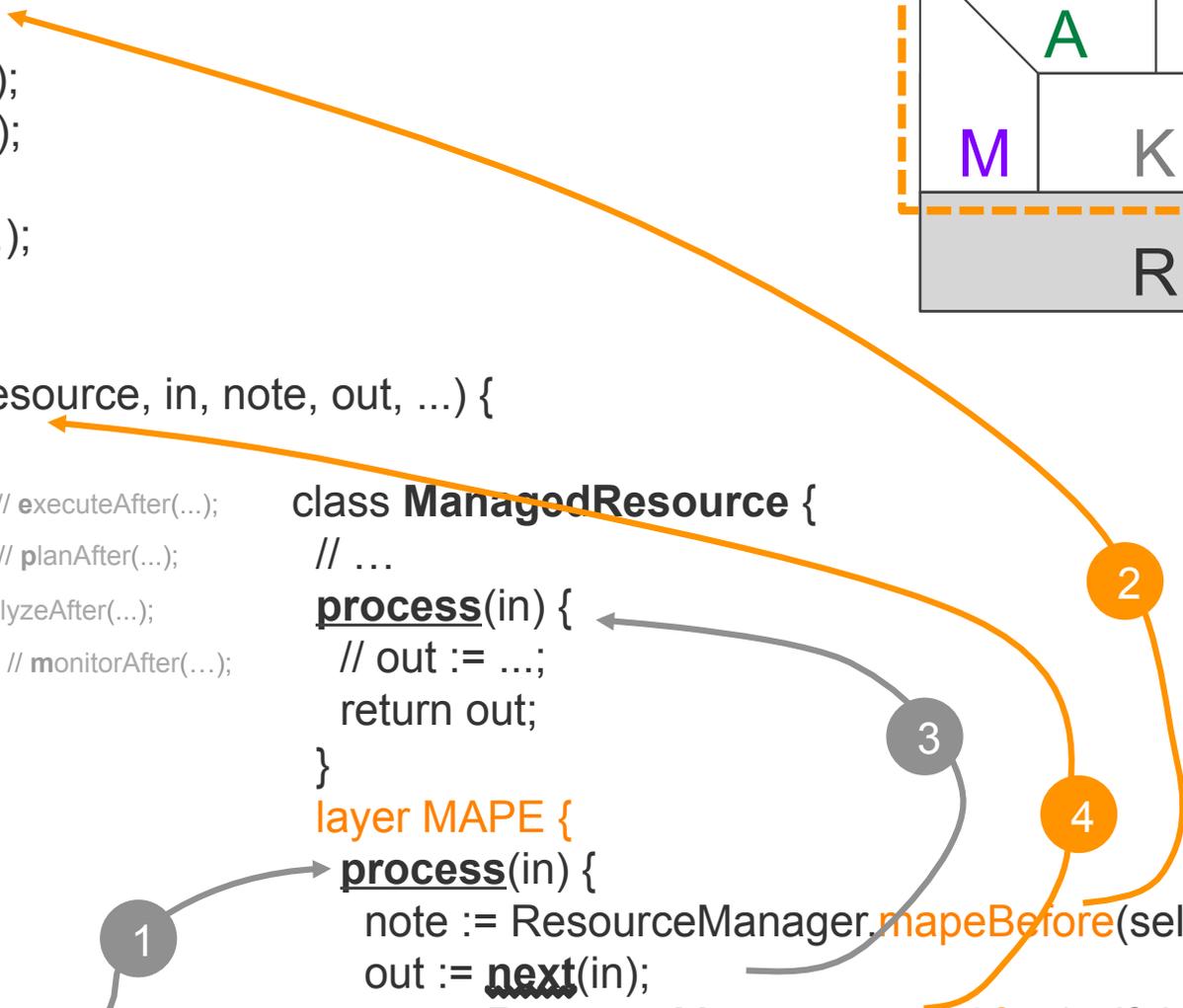
class ResourceManager {
  // ...
  layer MAPE { // not necessary
    static mapeBefore(resource, in, ...) {
      // note = ...;
      monitorBefore(...);
      analyzeBefore(...);
      planBefore(...);
      executeBefore(...);
      return note;
    }
    static mapeAfter(resource, in, note, out, ...) {
      // newOut = ...
      monitorAfter(...); // executeAfter(...);
      analyzeAfter(...); // planAfter(...);
      planAfter(...); // analyzeAfter(...);
      executeAfter(...); // monitorAfter(...);
      return newOut;
    }
  }}
  
```



```

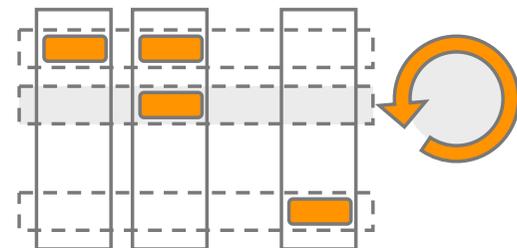
class ManagedResource {
  // ...
  process(in) {
    // out := ...;
    return out;
  }
  layer MAPE {
    process(in) {
      note := ResourceManager.mapeBefore(self, in, ...);
      out := next(in);
      out := ResourceManager.mapeAfter(self, in, note, out, ...);
      return out;
    }
  }}
  
```

A first sketch...



Acknowledgements

Pascal Costanza, Hidehiko Masuhara, Atsushi Igarashi, Michael Haupt, Malte Appeltauer, Michael Perscheid, Bastian Steinert, Jens Lincke, Marcel Taeumel, Tobias Pape, Tim Felgentreff, Robert Krahn, Carl Friedrich Bolz, Marcel Weiher, Hans Schippers, Tim Molderez, Oscar Nierstrasz, Shigeru Chiba, Hiroaki Inoue, Tobias Rho, Stefan Udo Hanenberg, Dick Gabriel, Dave Thomas, Gilad Bracha, Alan Kay, Dan Ingalls, Alan Borning, Jeff Eastman, Christopher Schuster, Christian Schubert, Gregor Schmidt, Stefan Lehmann, Matthias Springer, ...



Web References

- COP-related publications
 - HPI/SWA
<http://www.hpi.uni-potsdam.de/swa/publications/>
- Selected systems
 - JCop
<https://github.com/hpi-swa/JCop/>
 - ContextJS and Lively Webwerkstatt
<http://lively-kernel.org/repository/webwerkstatt/webwerkstatt.xhtml>
 - EventCJ
<http://prg.is.titech.ac.jp/projects/eventcj/>
 - Lively Kernel
<http://lively-kernel.org/>

