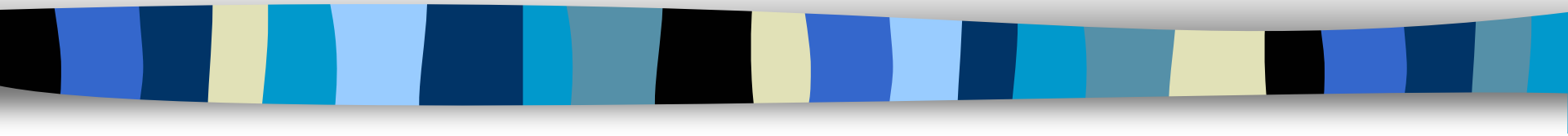
An AspectJ-enabled Eclipse Runtime Engine

- Demonstration at AOSD 04 -



Martin Lippert
lippert@acm.org
www.martinlippert.com

Motivation

- Use Eclipse 3.0 RCP to develop enterprise applications
- Use AspectJ to improve modularity

- **What happens if we want to use both techniques to develop applications?**
  - Especially to modularize cross-plugin pointcuts

Example

PLUGIN A

```
class Line {
  private Point p1, p2;

  Point getP1() { return p1; }
  Point getP2() { return p2; }

  void setP1(Point p1) {
    this.p1 = p1;
  }
  void setP2(Point p2) {
    this.p2 = p2;
  }
}
```

PLUGIN B

```
class Point {
  private int x = 0, y = 0;

  int getX() { return x; }
  int getY() { return y; }

  void setX(int x) {
    this.x = x;
  }
  void setY(int y) {
    this.y = y;
  }
}
```

PLUGIN C

```
aspect DisplayUpdating {

  pointcut move(FigureElement figElt):
    target(figElt) &&
    (call(void FigureElement.moveBy(int, int) ||
     call(void Line.setP1(Point))             ||
     call(void Line.setP2(Point))             ||
     call(void Point.setX(int))               ||
     call(void Point.setY(int)));


  after(FigureElement fe) returning: move(fe) {
    Display.update(fe);
  }
}
```

Example taken from the AspectJ Tutorial

Demo at AOSD 2004

Design Alternatives

- Recompile the complete system with AspectJ (ajc)
  - to weave an aspect into the whole system
- Weave the complete system once at startup-time
  - using the -injar option of AspectJ
- Weave (and re-weave) classes when necessary
  - using load-time bytecode weaving

I chose this alternative in order to be as compatible as possible with the Eclipse ideas

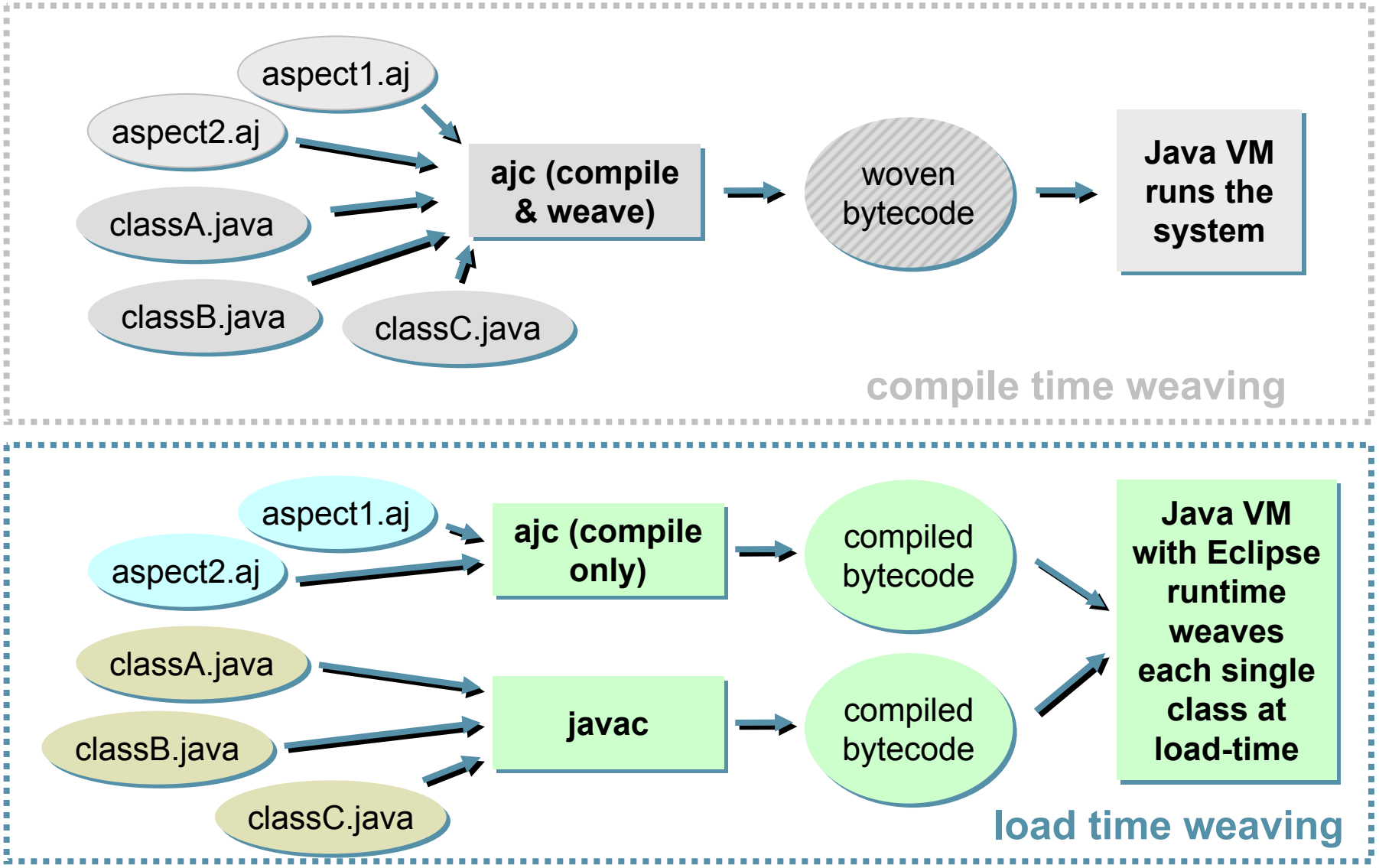
Solution: A load-time weaving runtime

- The basic idea:

  Let the Eclipse runtime weave aspects into plugins at *load-time*

Implementation

- load-time weaving -



Implementation

- aspect contribution -

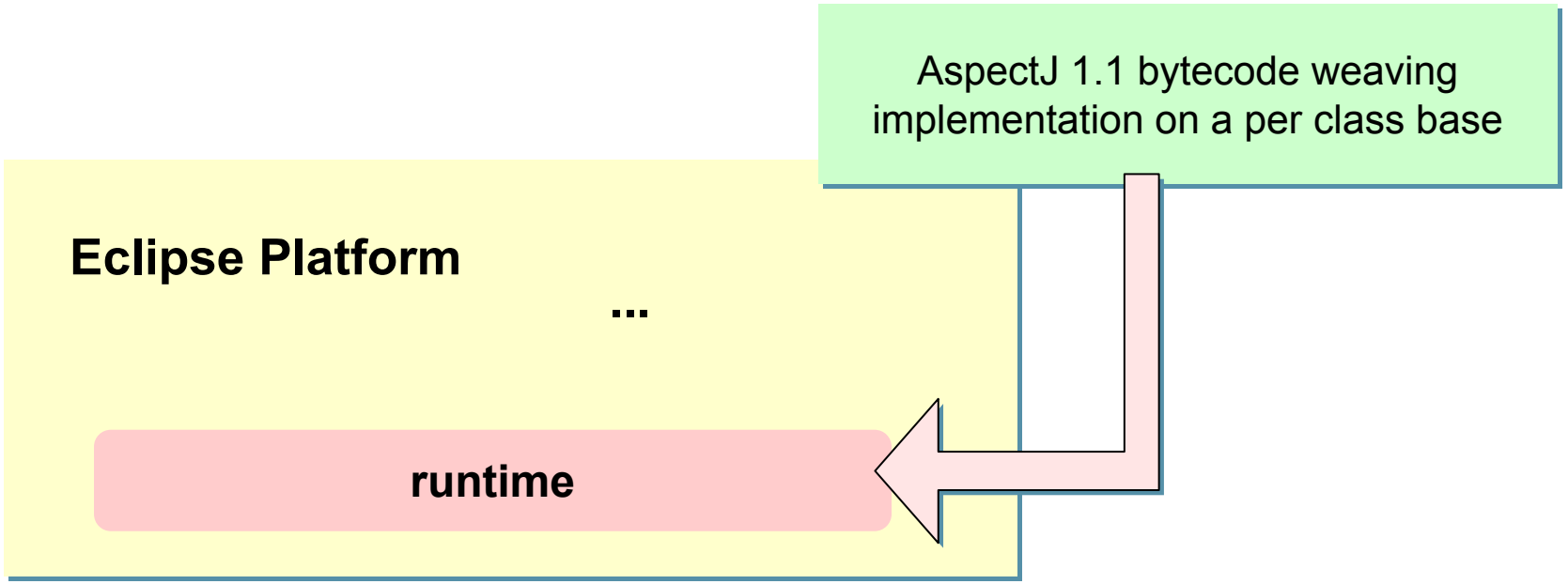
- Load-time weaving within the Eclipse runtime:
  - Weaving runtime needs to know all aspects that should be woven into
  - Solution:
    - Weaver plugin provides an extension point for aspects
    - Plugins can contribute aspects via extensions

```
<extension
      id="monitorruntime"
      name="monitorruntime"
      point="org.aspectj.weavingruntime.aspects">
   <aspect
      class="com.ibm.eclipse.monitor.aspect.MonitorAspect">
   </aspect>
</extension>
```

Implementation

- weaving inside the runtime -

- Problem:
  - load-time weaving has to happen at class-loading time
- Solution:
  - inject load-time bytecode modification into the Eclipse runtime



Implementation

- reusing weaver API from AspectJ -

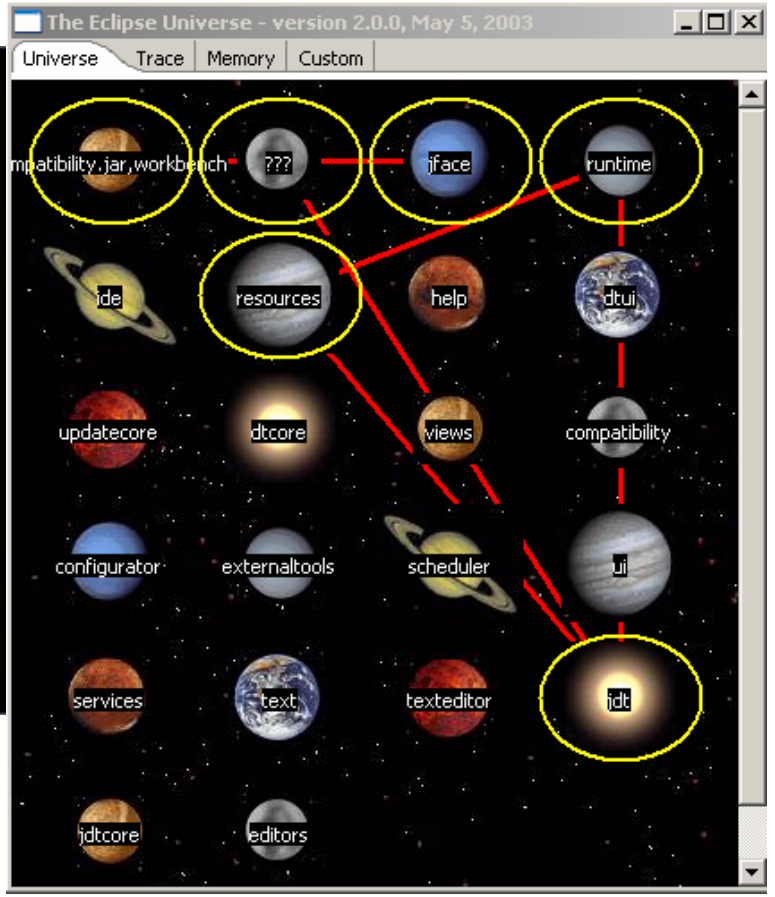
- Bytecode weaving plugin uses the weaver API from the AspectJ 1.1 implementation
  - No additional effort to implement aspect weaving
  - All improvements of AspectJ 1.2 can be re-used directly
  - Complete AspectJ language can be used

Hooking into the runtime

- Eclipse 2.1.x:
  - Modification of original runtime plugins needed
  - But still a fully compatible runtime
- Eclipse 3.0:
  - New OSGi-based runtime is a lot more flexible for this kind of extensions
  - Load-time bytecode modification can be implemented in separate plugin (via a specialized OSGi framework adaptor)
  - DynamicImport-Feature can be used to handle additional dependencies between plugins

Demo

- **Using the Eclipse IDE 3.0 itself as the application enhanced via aspects**



With special thanks to Chris Laffra for the Monitor plugin

More Use Cases

- within and beyond the Eclipse RCP -

- Analyzing - e.g.:
  - Find out where objects of a specific type are created
  - Find out where they are created depending on a specific control flow
- Enhancing - e.g.:
  - Do something every time a plugin is started, maybe depending on the control flow
- Modifying external libraries - e.g.:
  - Replace calls to the system class loader with calls to the class loader of the plugin
- . . .

Status of Work

- Implementation available for
  - Eclipse 2.1.2
  - 3.0M4 (old runtime)
  - 3.0M8 (new OSGi-based runtime)
- Features
  - Open Source
  - Load-time weaving for AspectJ 1.1 (and upcoming 1.2)
  - Caching for woven classes (to improve startup time)
- Availability
  - More information: www.martinlippert.com
  - If you are interested, please contact me: lippert@acm.org

The Next Steps

- Improvements
  - Performance
  - Footprint
  - Code refactorings
- Dynamic Plugins
  - New runtime features install/update/uninstall of plugins at runtime
  - What happens to aspects being installed/updated/uninstalled?
  - Solution: “run-time like” weaving for AspectJ
- Debugging
  - debugging within the PDE

Thank you for your attention !!!

- Questions highly welcome -

Special thanks to the Eclipse Runtime Team and the AspectJ-Team for their help and assistance implementing the prototype

Martin Lippert

lippert@acm.org

www.martinlippert.com