**The Vision:**
Using Eclipse Plugins and AspectJ Together

- Modularize cross-plugin concerns into separate plugins
- Build aspects into a platform to let other plugins follow the rules

Just let developers combine the benefits from both worlds without limitations!!!

**The Problem:**
Separate Compilation vs. Aspect Weaving

- AspectJ needs to weave the system to let aspects work correctly
- You would need to recompile the whole system when new aspects come in, old aspects are deleted or existing aspects change
- You would not be able to plug in new plugins into an existing system without recompilation

This cuts off major features of the Eclipse plugin technology

**The Challenge:**
Dynamic Plugins and Aspects

- OSGi-based runtime allows plugins to be (un-)installed at runtime
- What happens if aspect-promoting plugins are (un-)installed at runtime???

Possible solution:
- AJEER has to take care to update active plugins at runtime (update mechanism of OSGi kernel)
- Therefore, AJEER needs to keep track of aspect dependencies and possible targets of aspects

AJEER: Load-Time Aspect Weaving for the Eclipse Platform
Enabling programmers to combine Eclipse plugin technology and AspectJ
Martin Lippert (lippert@acm.org) - Download: http://www.martinlippert.org/

**The Solution:**
AJEER: An AspectJ-Enabled Eclipse Runtime

Features weaving of AspectJ aspects into existing plugins at class-loading time (including dependency management)

- Standard Eclipse plugins can add new aspects to the system via a new extension point (aspect-promotion)
- Already existing or new plugins do not need to be recompiled
- Separate compilation for plugins still possible
- Open issues: performance and footprint of the weaving process (but weaving performance improves and AJEER implements caching of woven plugins)

**Example 1:** The Eclipse Monitor

**Example 2:** The Zipper System for Replicated Application Sharing

**Example 3:** Parallax

Enabling programmers to combine Eclipse plugin technology and AspectJ

Martin Lippert (lippert@acm.org) - Download: http://www.martinlippert.org/

**Example 1:** The Eclipse Monitor

- AJEER-based implementation of Chris Laffra's Eclipse-Monitor
- Shows the internal behavior of Eclipse (method calls, object creations, plugin communication)

**Example 2:** The Zipper System for Replicated Application Sharing

- Replicated Application Sharing for Eclipse
  - currently works with text-based editors
  - GEF in progress
- Uses Aspects for catching events, code archeology (which events to catch?), and missing API workarounds
  http://www.research.ibm.com/zipper
- Steven Rohall
  IBM T.J.Watson Research Center

**Example 3:** Parallax

- MDA Eclipse Support for Addressing Middleware-Specific Crosscutting Concerns Based on AspectJ
- http://parallax.lgl.epfl.ch/
  - Raul Silaghi [rsilaghi@acm.org]
  - Software Engineering Laboratory
  - Swiss Federal Institute of Technology in Lausanne (EPFL)

Get a Demo!
Just ask for it!!!