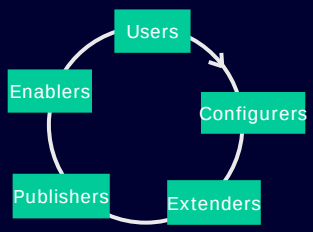
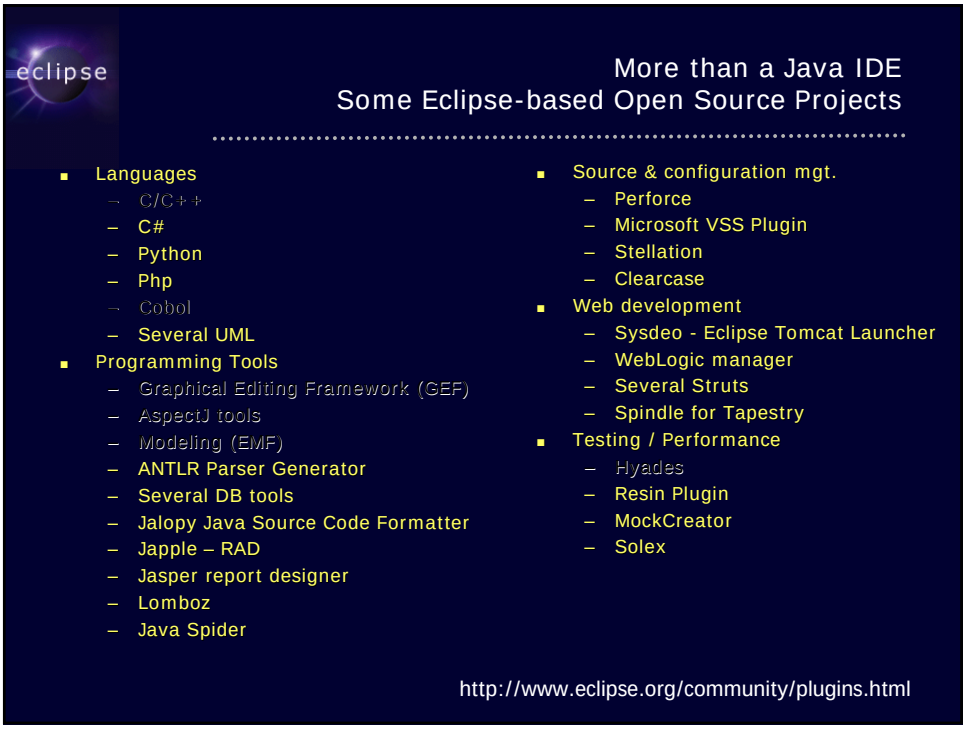
Eclipse
more than a Java IDE

erich_gamma@ch.ibm.com
Eclipse Java Development Tools Lead
Eclipse PMC Member



What is Eclipse?


eclipse
What is Eclipse?
“Eclipse is a kind of universal tool platform - an open extensible IDE for anything and nothing in particular. “
Eclipse is more than a Java IDE…


eclipse
More than a Java IDE
Some Eclipse-based Open Source Projects
Languages
C/C++
C#
Python
Php
Cobol
Several UML
Programming Tools
Graphical Editing Framework (GEF)
AspectJ tools
Modeling (EMF)
ANTLR Parser Generator
Several DB tools
Jalopy Java Source Code Formatter
Japple – RAD
Jasper report designer
Lomboz
Java Spider
Source & configuration mgt.
Perforce
Microsoft VSS Plugin
Stellation
Clearcase
Web development
Sysdeo - Eclipse Tomcat Launcher
WebLogic manager
Several Struts
Spindle for Tapestry
Testing / Performance
Hyades
Resin Plugin
MockCreator
Solex
http://www.eclipse.org/community/plugins.html

What is Eclipse?

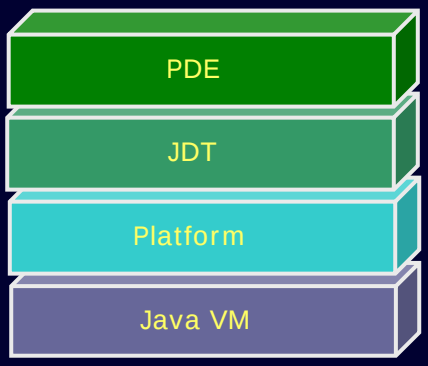
- Eclipse is a universal platform for integrating development tools
- Open, extensible architecture based on plug-ins

Plug-in development environment

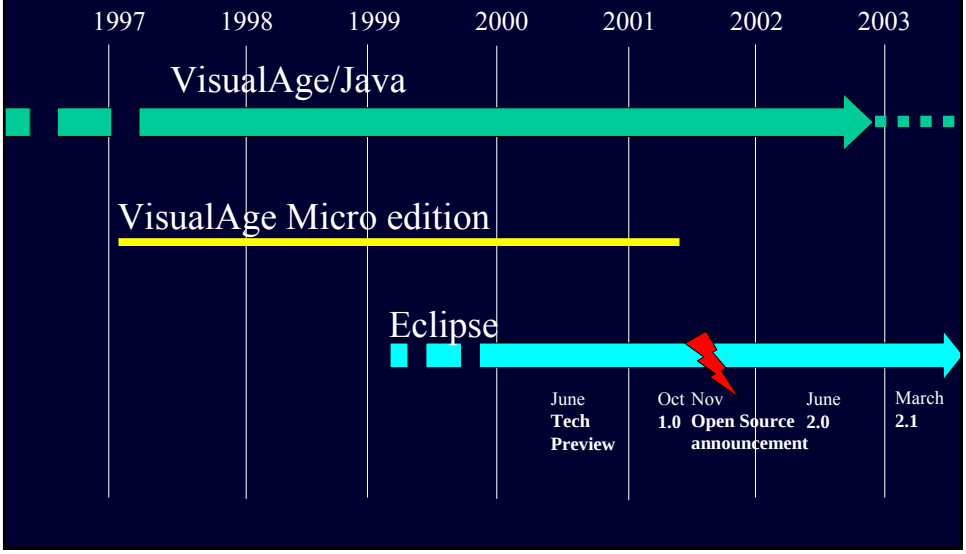
Java development tools

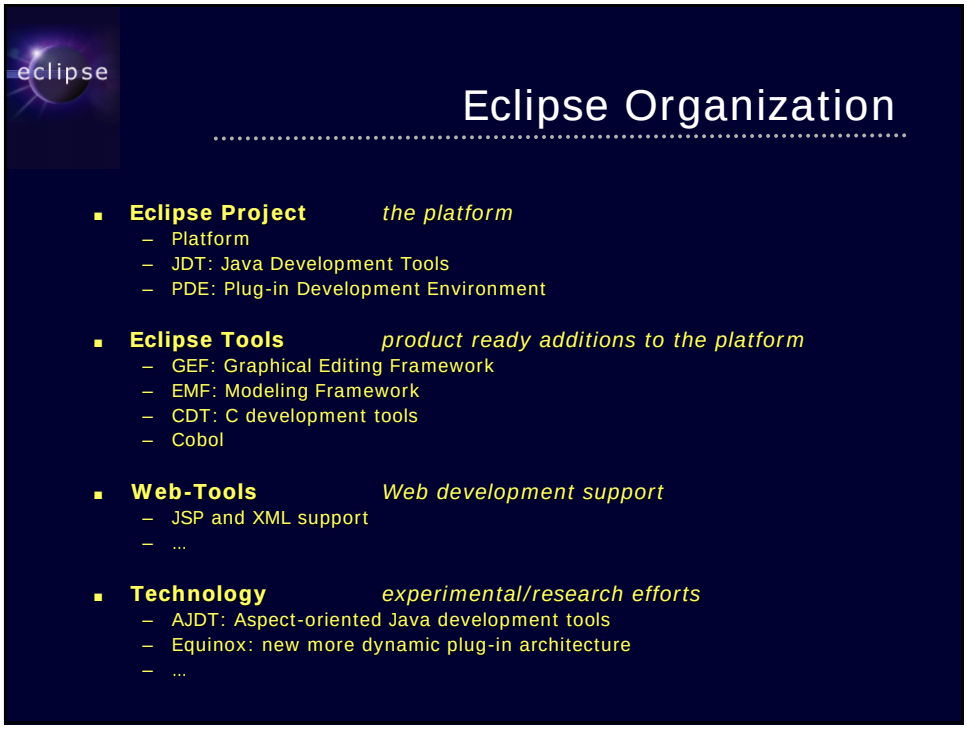
Eclipse Platform

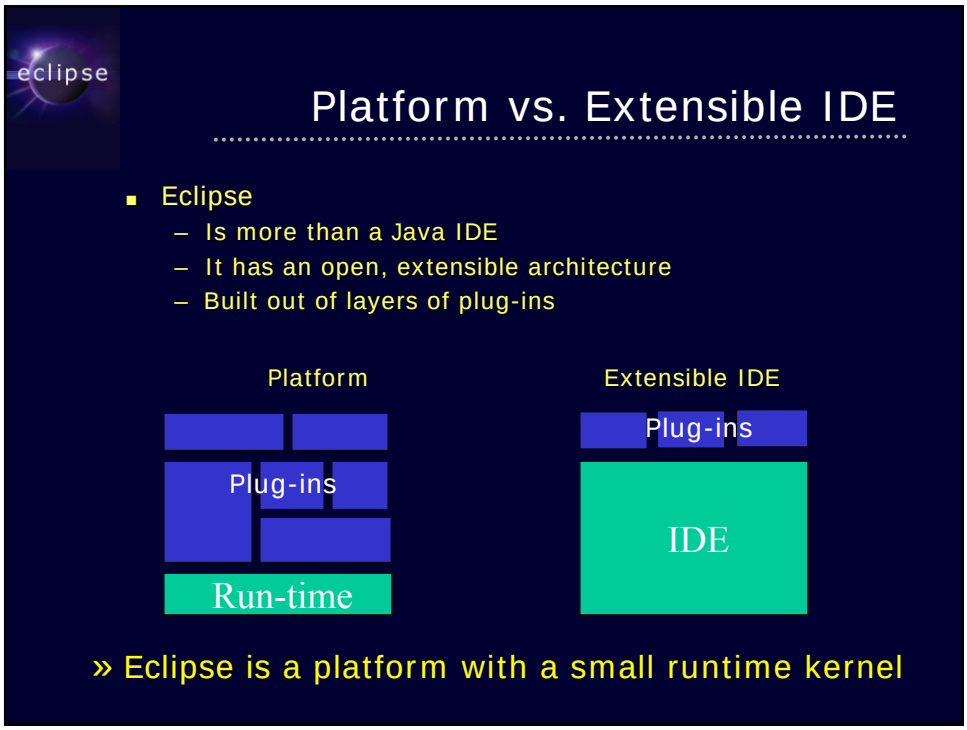
Standard Java2 Virtual Machine

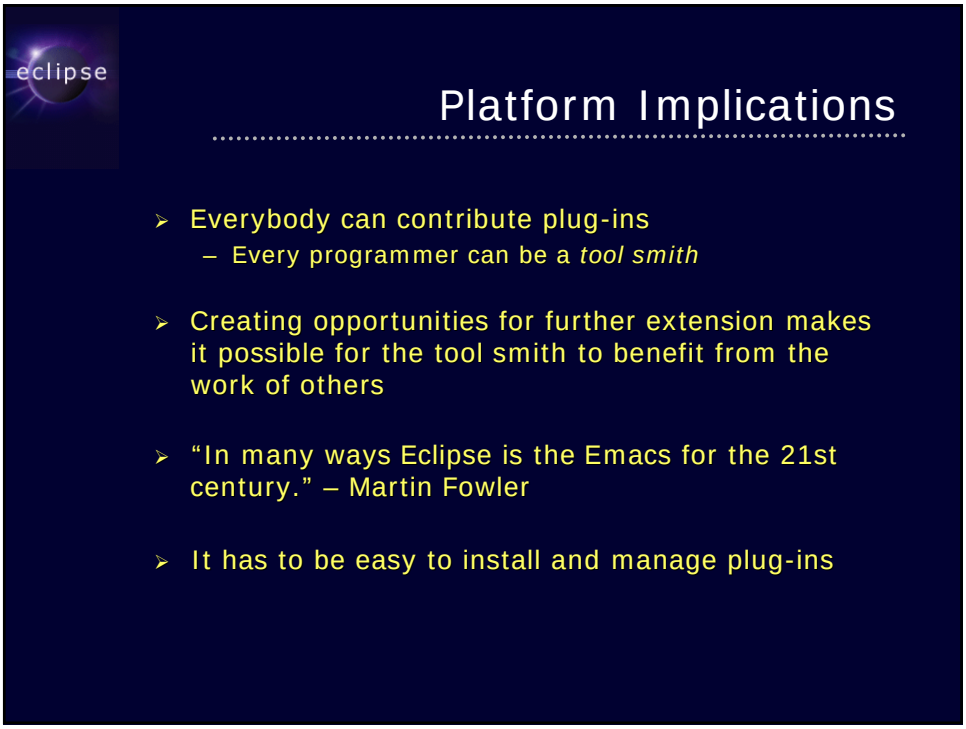


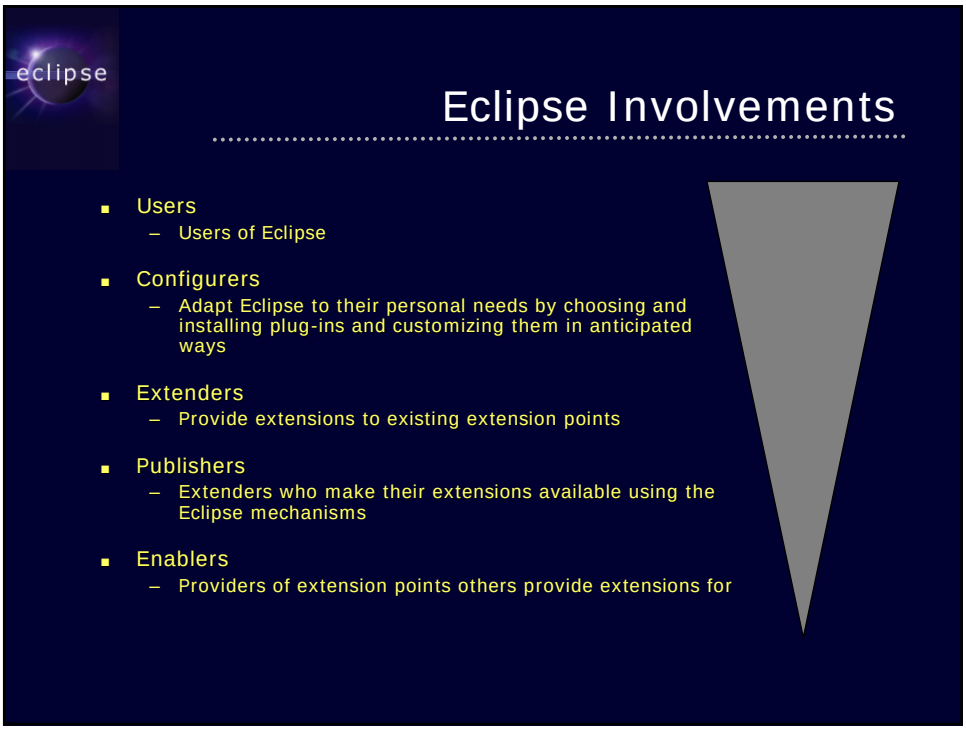
The Way to Eclipse




eclipse
Eclipse Organization
Eclipse Project the platform
Platform
JDT: Java Development Tools
PDE: Plug-in Development Environment
Eclipse Tools product ready additions to the platform
GEF: Graphical Editing Framework
EMF: Modeling Framework
CDT: C development tools
Cobol
Web-Tools Web development support
JSP and XML support
…
Technology experimental/research efforts
AJDT: Aspect-oriented Java development tools
Equinox: new more dynamic plug-in architecture
…


eclipse
Platform vs. Extensible IDE
Eclipse
Is more than a Java IDE
It has an open, extensible architecture
Built out of layers of plug-ins
Platform
Plug-ins
Run-time
Extensible IDE
Plug-ins
IDE
» Eclipse is a platform with a small runtime kernel


eclipse
Platform Implications
Everybody can contribute plug-ins
– Every programmer can be a *tool smith*
Creating opportunities for further extension makes it possible for the tool smith to benefit from the work of others
"In many ways Eclipse is the Emacs for the 21st century." – Martin Fowler
It has to be easy to install and manage plug-ins

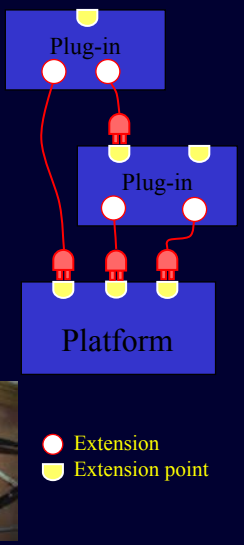

eclipse
Eclipse Involvements
Users
– Users of Eclipse
Configurers
– Adapt Eclipse to their personal needs by choosing and installing plug-ins and customizing them in anticipated ways
Extenders
– Provide extensions to existing extension points
Publishers
– Extenders who make their extensions available using the Eclipse mechanisms
Enablers
– Providers of extension points others provide extensions for

Plug-in Goals...

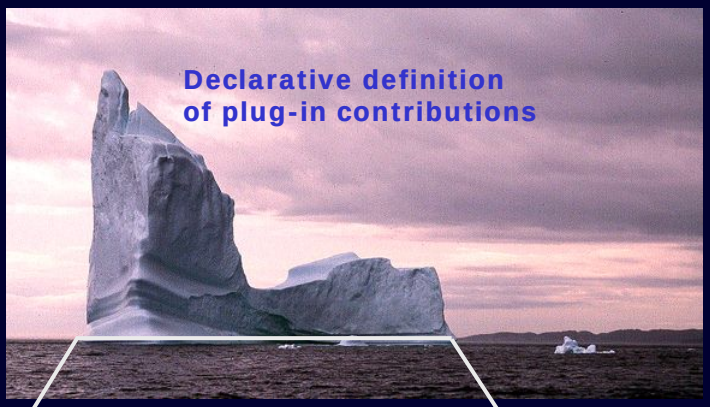
- Easy to develop
  - Java Development Tools + Plug-in development environment
- Scale-up to hundreds of installed plug-ins
  - the problem is start-up time...
  - Eclipse consists of 67 plug-ins, WSAD IE is even larger > 500 plug-ins
  - ➢ lazy loading
- Easy to discover, install, and update
  - install/update support
- Easy to manage an installation
  - managed configurations

Eclipse Plug-in Architecture

- **Plug-in** – set of contributions
  - Smallest unit of Eclipse functionality
  - Big example: HTML editor
  - Small example: Action to count lines
- **Extension point** – named entity for collecting contributions
  - Example: extension point for adding software metrics
- **Extension** – a contribution
  - Example: a specific metric



Tip of iceberg



Implementation
of plug-in contributions

- startup time: O(#used plug-ins), not O(# installed plug-ins)

Extension configuration in XML

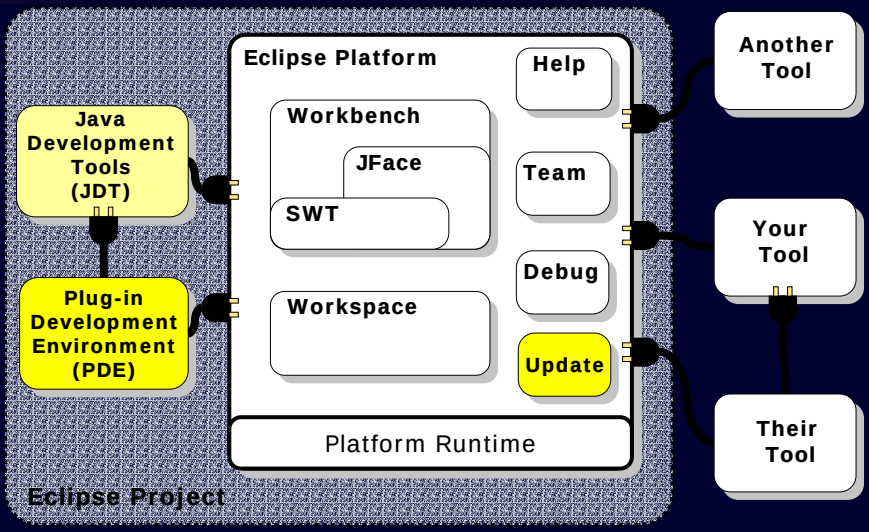
```
<extension-point name="Example" id="example"schema="schema/example.exsd"/>
```

Extension point definition

```
 <extension point="org.eclipse.ui.preferencepages">
  <page id="com.example.myprefpage"
    icon="icons/image.gif"
    title="My title"
    class="com.example.mywizard">
  </page>
</extension>
</plugin>
```

Extension contribution

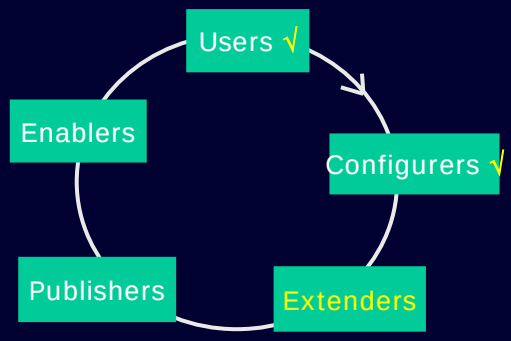
Eclipse Supports Plug-in Development



Monkey See/Monkey Do

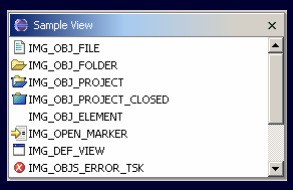
- Eclipse Trio
  - tools
  - published APIs
  - Open Source → source access
- Learn-by-example
  - browse existing code
  - PDE
  - JSpider

The Contribution Cycle



Extender: Contribute an Icon View

- Goal: a plug-in to view the standard Eclipse images
- Steps:
  - read extension point specifications
  - use Plug-in Development Tools to create a plug-in project and to declare the extension
  - use the Java Development Tools to implement the extension

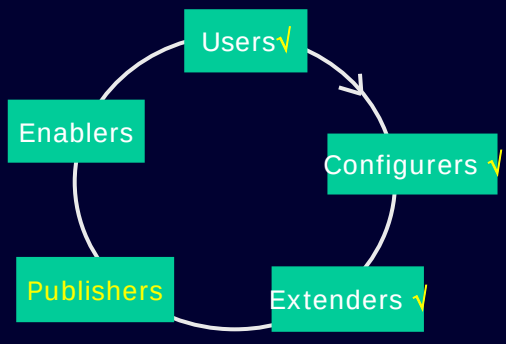


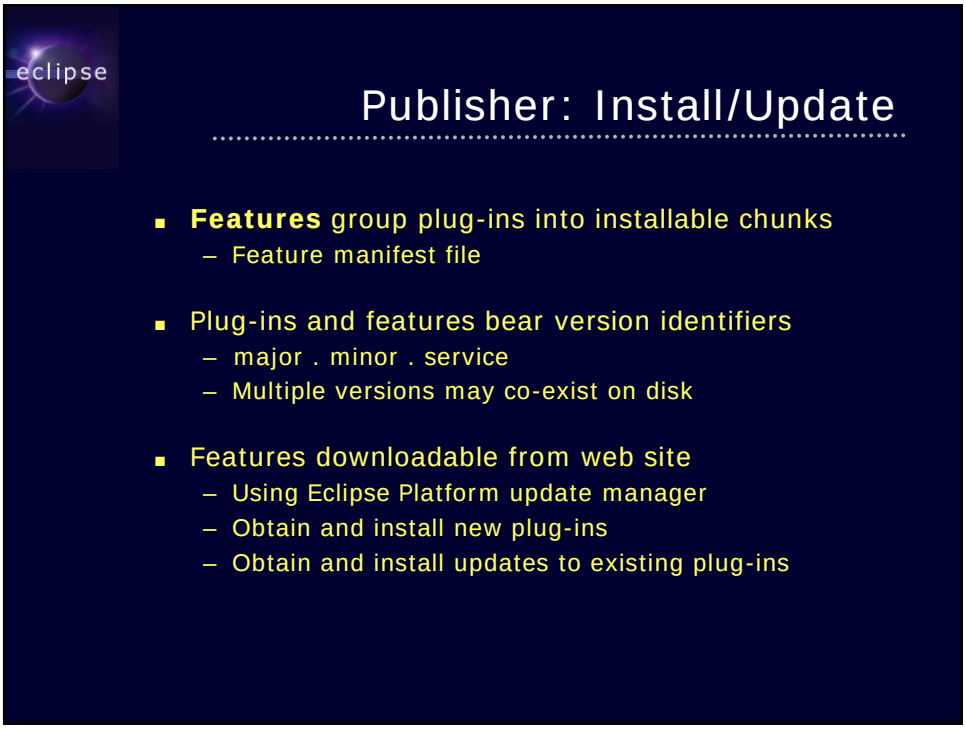
House Keeping Rules for Extenders *

- **Conformance Rule**: Contributions must conform to expected interfaces
- **Relevance Rule**: only contribute when you can successfully operate
  - you are not the only contributor…
- **Sharing Rule**: Add, don't replace
- **Integration Rule**: Integrate, don't separate
- **Responsibility Rule**: Clearly identify your plug-in as the source of problems
- **Strata Rule:** Separate language-neutral functionality from language-specific functionality and separate core functionality from UI functionality.
- **Program To API Contract Rule:** Check and program to the Eclipse API contract.

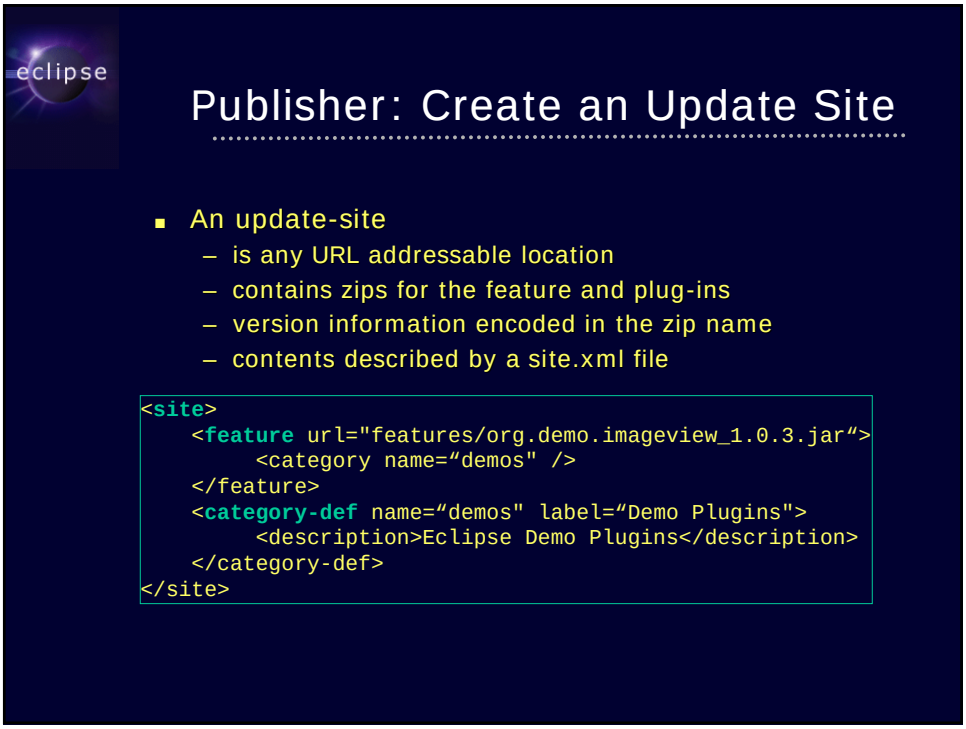
*Erich Gamma, Kent Beck - Contributing to Eclipse: Practices, Plug-Ins, Patterns

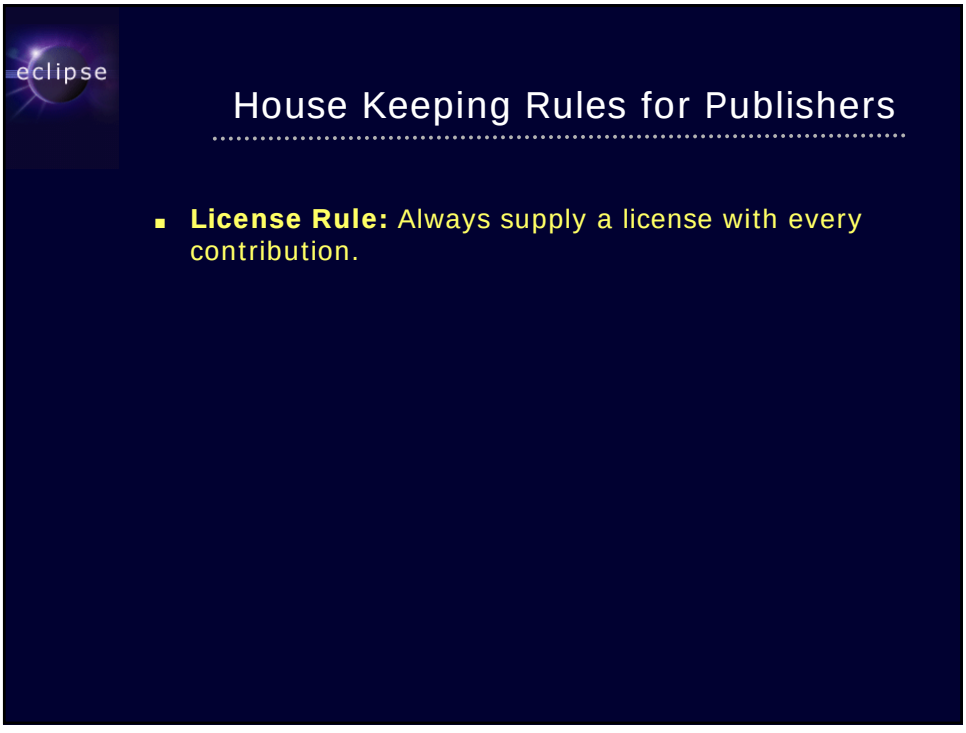
The Contribution Cycle



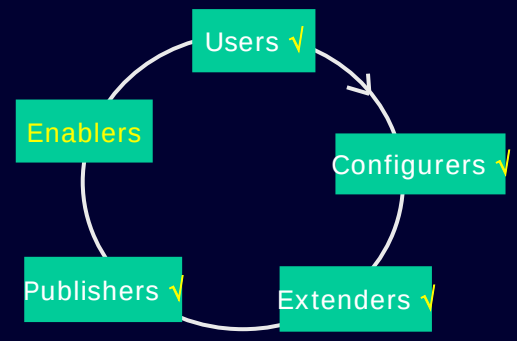

eclipse
Publisher: Install/Update
Features group plug-ins into installable chunks
– Feature manifest file
Plug-ins and features bear version identifiers
– major . minor . service
– Multiple versions may co-exist on disk
Features downloadable from web site
– Using Eclipse Platform update manager
– Obtain and install new plug-ins
– Obtain and install updates to existing plug-ins


eclipse
Publisher: Create a Feature
Feature describes
– Contained plug-ins and their versions
– Pre-requisite plug-ins for the feature
<feature
id="org.demo.imageviewfeature“ version="1.0.0">
<requires>
<import plugin="org.eclipse.core.resources"/>
<import plugin="org.eclipse.ui"/>
</requires>
<plugin
id="org.demo.imageview"
download-size="0"
install-size="0"
version="1.0.0"/>
</feature>

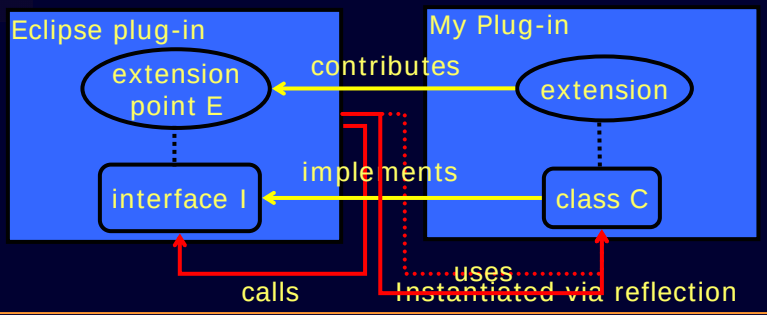

eclipse
Publisher: Create an Update Site
An update-site
is any URL addressable location
contains zips for the feature and plug-ins
version information encoded in the zip name
contents described by a site.xml file
<site>
<feature url="features/org.demo.imageview_1.0.3.jar“>
<category name=“demos" />
</feature>
<category-def name=“demos" label=“Demo Plugins">
<description>Eclipse Demo Plugins</description>
</category-def>
</site>


eclipse
House Keeping Rules for Publishers
License Rule: Always supply a license with every contribution.

The Contribution Cycle



Enabler: Invite Others to Contribute



- Declares extension point:
  `<extension-point id="imageFilters"/>`

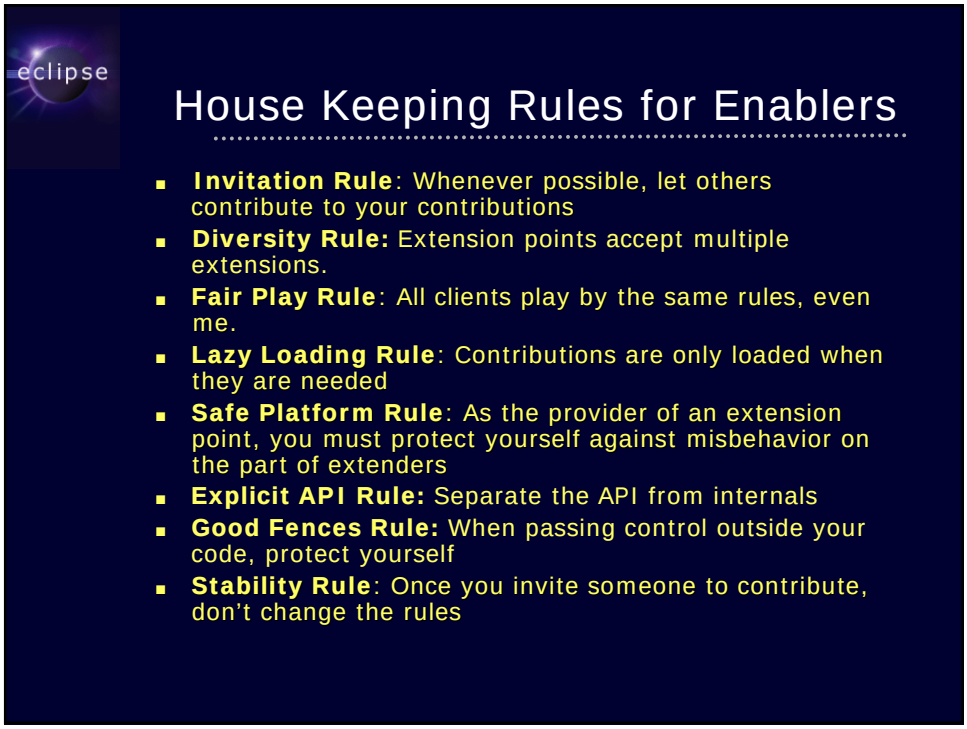
plugin.xml

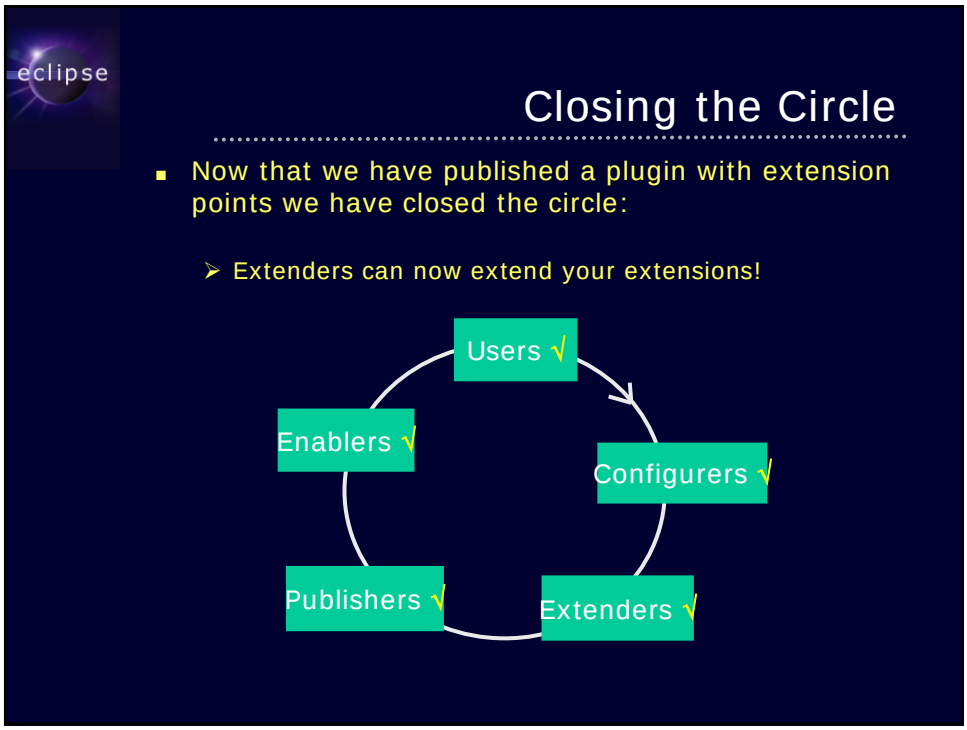
- Contributes extension:
  `<extension point="….imageFilters"/>`
  `…class=".GreyFilter"`
  `</extension>`

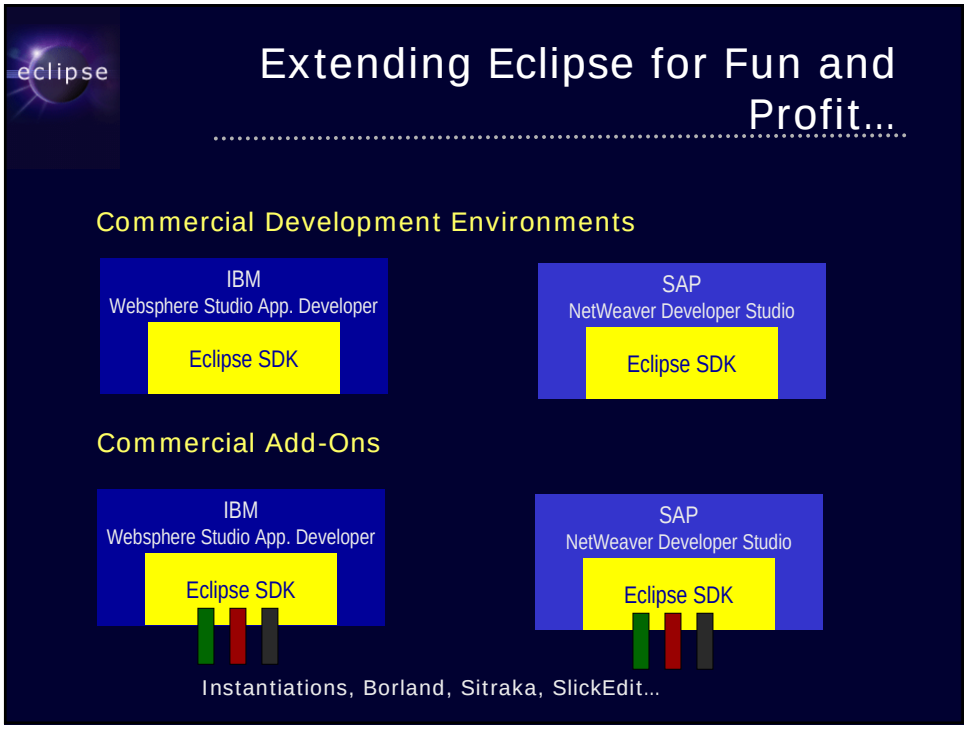
- Declares interface:
  `interface IImageFilter {`
  `Image filter(Image image);`
- `}`

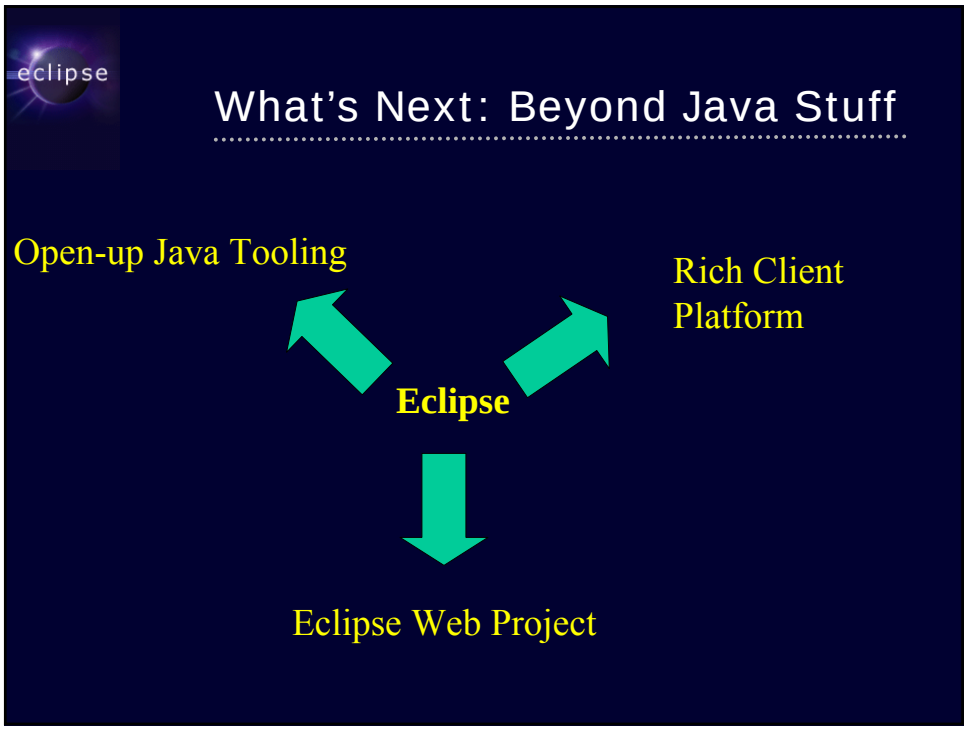
Java code

- Implements interface:
  `class GreyFilter implements IImageFilter {`
  `}`


eclipse
House Keeping Rules for Enablers
Invitation Rule: Whenever possible, let others contribute to your contributions
Diversity Rule: Extension points accept multiple extensions.
Fair Play Rule: All clients play by the same rules, even me.
Lazy Loading Rule: Contributions are only loaded when they are needed
Safe Platform Rule: As the provider of an extension point, you must protect yourself against misbehavior on the part of extenders
Explicit API Rule: Separate the API from internals
Good Fences Rule: When passing control outside your code, protect yourself
Stability Rule: Once you invite someone to contribute, don't change the rules


eclipse
Closing the Circle
Now that we have published a plugin with extension points we have closed the circle:
Extenders can now extend your extensions!
Users √
Configurers √
Extenders √
Publishers √
Enablers √


eclipse
Extending Eclipse for Fun and Profit…
Commercial Development Environments
IBM
Websphere Studio App. Developer
Eclipse SDK
SAP
NetWeaver Developer Studio
Eclipse SDK
Commercial Add-Ons
IBM
Websphere Studio App. Developer
Eclipse SDK
SAP
NetWeaver Developer Studio
Eclipse SDK
Instantiations, Borland, Sitraka, SlickEdit…

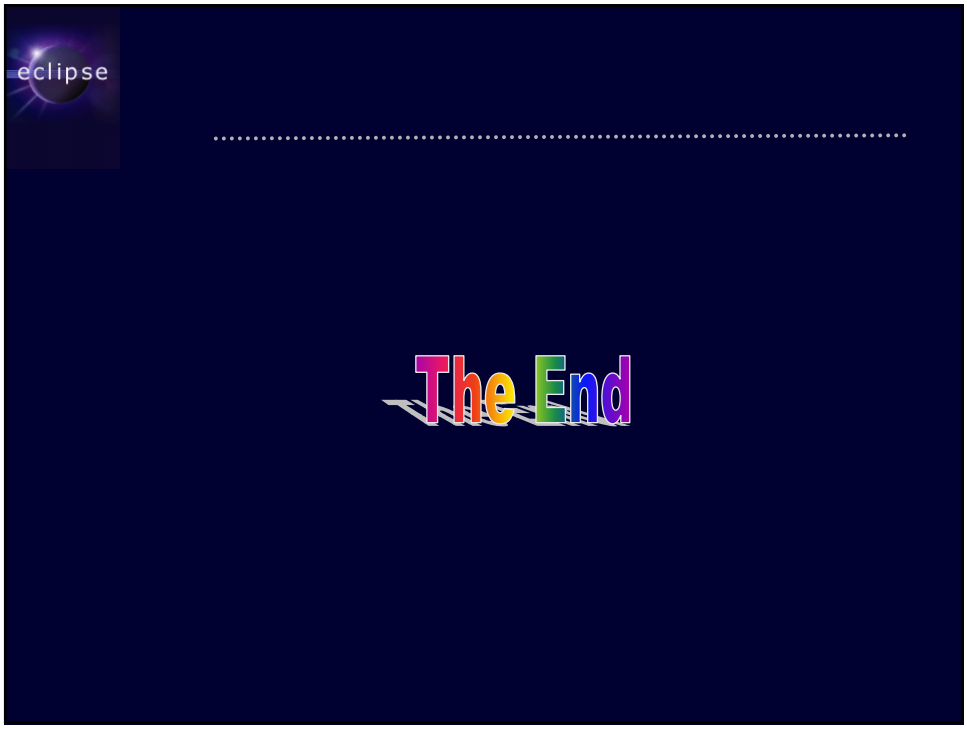

eclipse
What's Next: Beyond Java Stuff
Open-up Java Tooling
Rich Client Platform
Eclipse
Eclipse Web Project

Eclipse 3.0 Themes

- Platform
  - improved user experience
    - UI scalability in the face of tons of contributions
    - out of the box experience
  - responsive UI
    - background activities
  - rich client platform
    - generalize Eclipse into a platform for building non-IDE apps
- Java Development Tools
  - open-up for other Java family members
  - improved user experience
    - navigation
  - digesting Tiger

Summary

- All functionality is provided by plug-ins and fragments
  - Includes all aspects of Eclipse Platform itself
- Contributions are made via extension points
  - Extensions are created lazily
- Plug-ins and fragments are packaged into separately installable features
  - Downloadable
- PDE and JDT turn Eclipse into the development environment to develop Eclipse plug-ins


eclipse
The End