What you always wanted to know about

Developing for Eclipse/CSS

but were afraid to ask

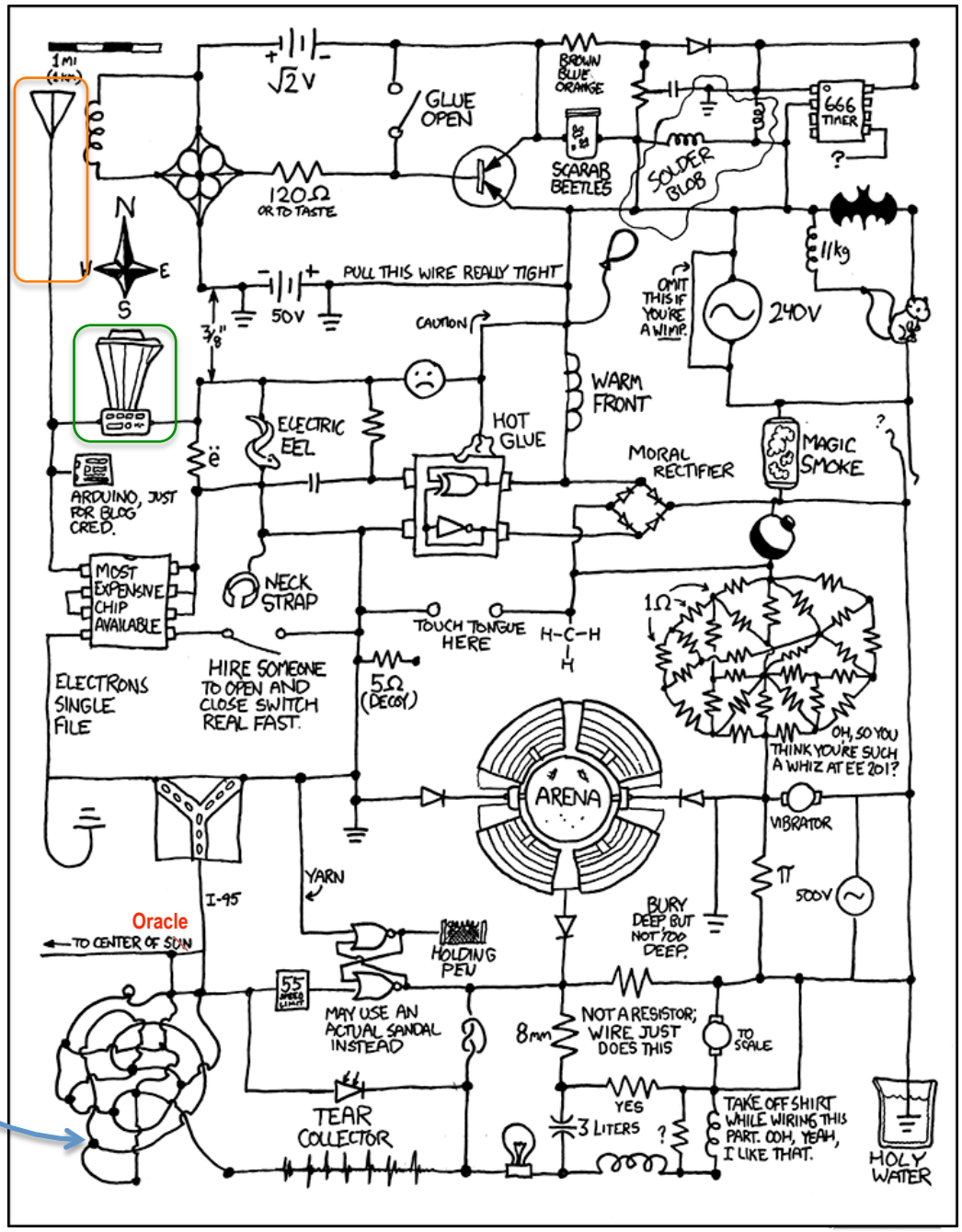
Kay Kasemir

June 2010

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Outline

- **Note:**
  - **Diagram shows Eclipse 3.5.2 and SNS CSS 2.0.1!**
  - **DESY CSS 1.2.0 allows optional Tine support here:**
  - **Also:**
    - **Your accelerator is different**
    - **Wireless = Security Risk**



http://xkcd.com/730/

Audience

- Familiar with Java?
  - Can list differences between Java 5 and 6?

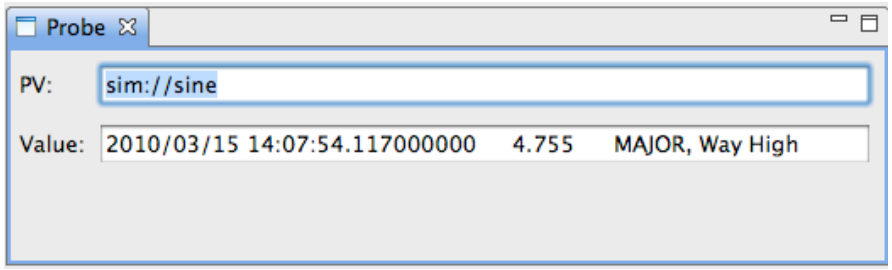
- Have used Eclipse as an IDE?

  - … for test-driven development?
  - … to refactor code?
- Have written Eclipse plug-ins, RCP apps?

  - … including CSS?
- Worry about bug 300500?

This Talk

- Eclipse as an IDE
  - Develop "Formula", test-driven, with refactoring
- Plug-ins
  - Package Formula as plug-in
- Extension Points
  - Build simplistic "NIOC"
- CSS Plug-ins and Extension Points
  - Build "Probe"
- Avoid bug 300500



Basic Eclipse Setup

- Use Sun Java SDK (not JRE, not GCJ)

- Download “Eclipse for RCP/Plug-In Developers”

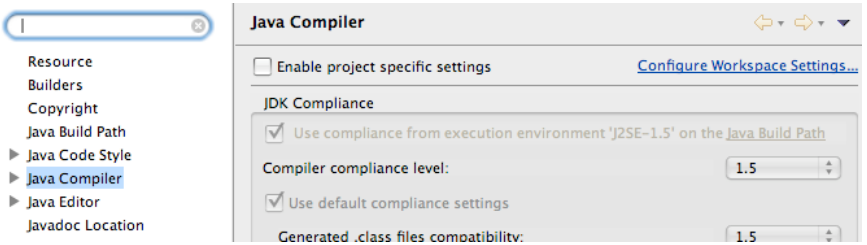
- Suggested Eclipse Preferences
  - Java, Installed JREs:
    - Set default to your SDK
  - Java, Compiler: Compiler compliance level:
    - “1.5” will create code that works for Java 1.5 and 1.6.
  - Java, Compiler, Errors/Warnings:
    - Enable as many as possible. Warnings help avoid errors

“Formula” Project

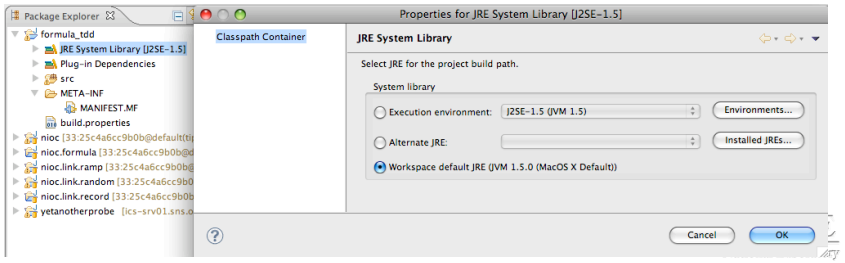
- Menu File, New Project
  - Plug-in Development, Plug-in Project
    - Name: “Formula”
    - Use defaults
    - MANIFEST.MF, “Dependencies”: org.junit4

Unless a plugin has special requirements, I suggest to use the workspace settings (and they in turn target java 1.5 and higher)

- Right-click on new project in Package Explorer
  - Properties, Java Compiler:
    Uncheck
    “Enable Project specific settings”



- Right-click on project’s
  “JRE System Library”
  in Package Explorer
  - Properties: Select “Workspace default”



“Formula” Library

- What should it do?

  2+3 *x*atan(3/2)*
- API?

  Expression.evaluate(“2+3”)
- Did you test it?
  - Well, back when I wrote it.
- Does it still work after Fred *“improved”* it?
  - Maybe…
- Do you have examples for using it?
  - You mean a simple, standalone example?

Test Driven Development

- Write the test first!
  - Clarify scope
  - Brainstorm API
- Eclipse helps
  - Skeleton implementation
    - class Formula
    - double eval()
  - Refactoring
    - Rename to “Expression”
    - Extract Method

```java
/** JUnit test of the Formula
 *  @author Kay Kasemir
 */
public class FormulaTest
{
    @Test
    public void testBasicFormula()
    {
        // Should it work like this?
        // double result = Formula.eval("2+3");

        // Or like this?
        final Formula formula = new Formula("2+3");
        double result = formula.eval();

        // In any case:
        assertEquals(5.0, result, 0.001);
    }


    @Test
    public void testAdvancedFormula()
    {
        final Formula formula = new Formula("2-3");
        assertEquals(-1.0, formula.eval(), 0.001);
    }
}
```

http://www.junit.org: @Test, assertEquals(…), Run As/JUnit

Google Eclipse TDD, http://vimeo.com/10569751

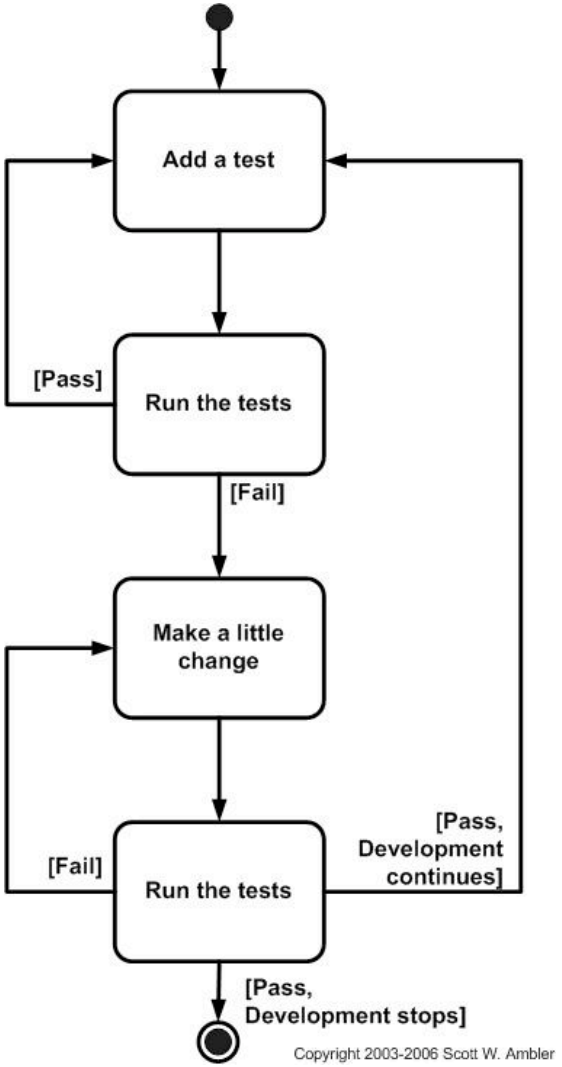
formula_tdd

Test Driven Development

Red, Green, Refactor

- Once Test passes:
  - Done!
- Plus
  - Can re-run after “improvements”
  - Extend test when new requirements arrive
    - Variable names that start with “51…”
  - Tests serve as developer examples

For more about TDD, Agile Methods, why the waterfall model doesn’t work: Scott Ambler, Martin Fowler



nioc.formula

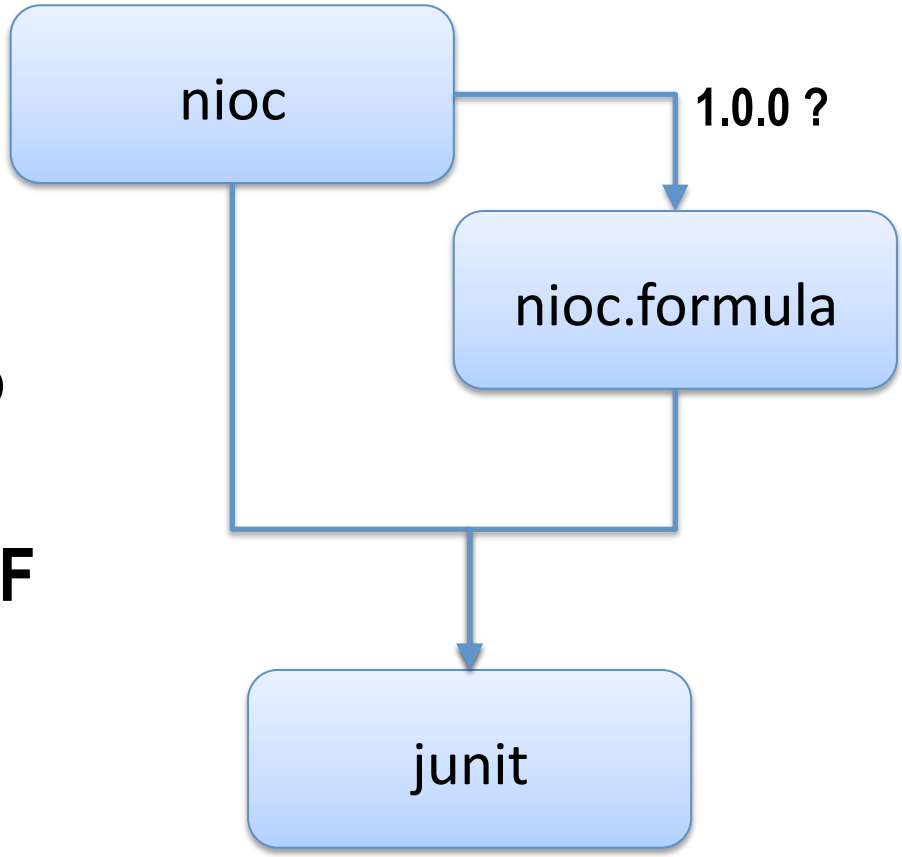
Eclipse Plug-Ins: Packaging

- Jar(=ZIP) file or directory for grouping stuff
  - Java Sources & Classes
  - HTML files, Images, …
- Extra MANIFEST.MF entries for
  - Access restrictions
    - nioc.formula exports the Formula, but not the tests & internals
  - Class path handling
    - Define plug-in dependencies instead of editing CLASSPATH
  - Versioned dependencies

nioc, nioc.formula: MANIFEST.MF

Equinox Runtime

- Loads Plug-ins and their dependencies as needed ("lazy")
- Adds exported packages to CLASSPATH
- Handles extra MANIFEST.MF entries for Live Cycle
  - "Activator" start/stop



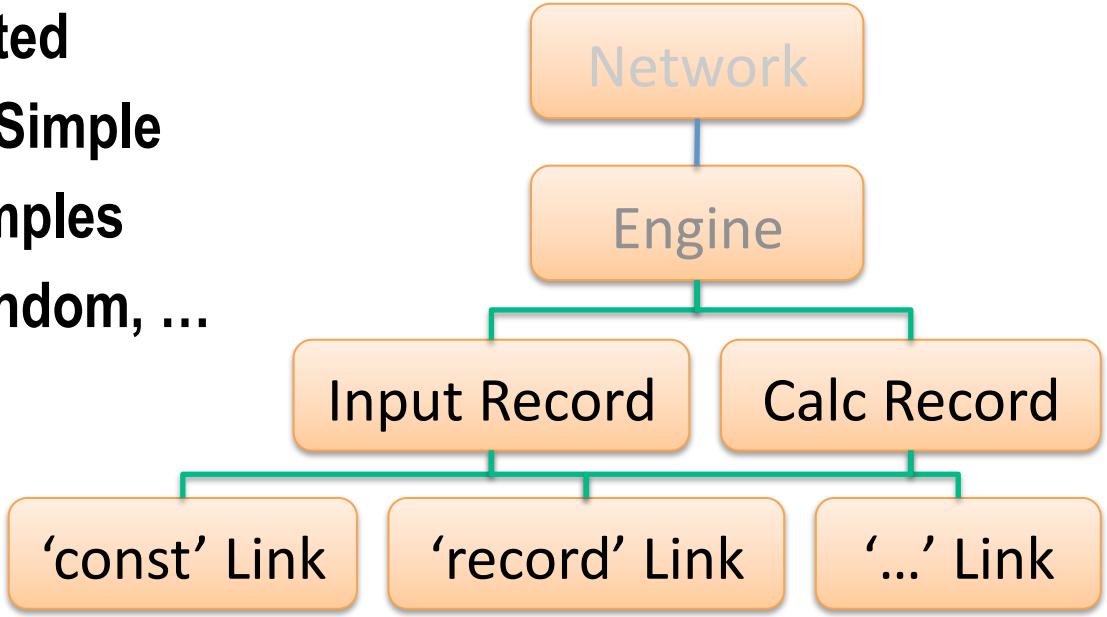
“NIOC” Project

```
<!-- Example NIOC Database -->
<records>

  <record type="input" name="the_answer">
    <description>Input example, reads constant</description>
    <input type="const">42</input>
  </record>

  <record type="input" name="noise">
    <description>Input example, reads random number</description>
    <input type="random"/>
  </record>

  <record type="input" name="rampup">
    <description>Input example, reads 'ramp'</description>
    <input type="ramp"/>
  </record>

  <record type="input" name="copy">
    <description>Input example, reads another record</description>
    <input type="record">noise</input>
  </record>

  <record type="calc" name="twice">
    <description>Calc example</description>
    <input name="scale_factor" type="const">2</input>
    <input name="other_record" type="record">rampup</input>
    <formula>1.0 * scale_factor * other_record + (2 + 3 - 5)</formula>
  </record>

</records>
```

nioc: example.xml

What do we need for an IOC?

- Network protocol – Omitted
- Scanning Engine – Very Simple
- Records – 2 Simple Examples
- Links – const, record, random, …
- Shell



Emphasis:

Extension Points to connect 'Records', 'Links'

Console as shell.

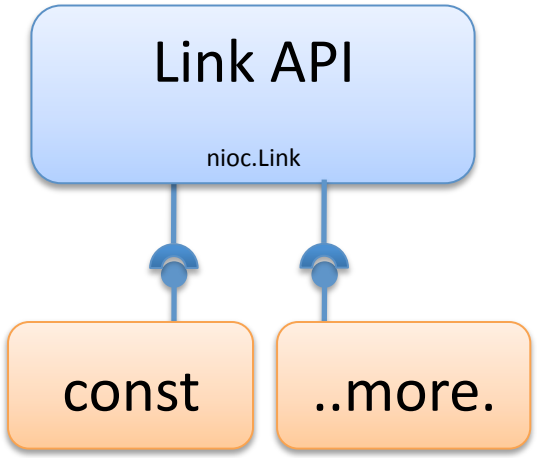
Links, Records

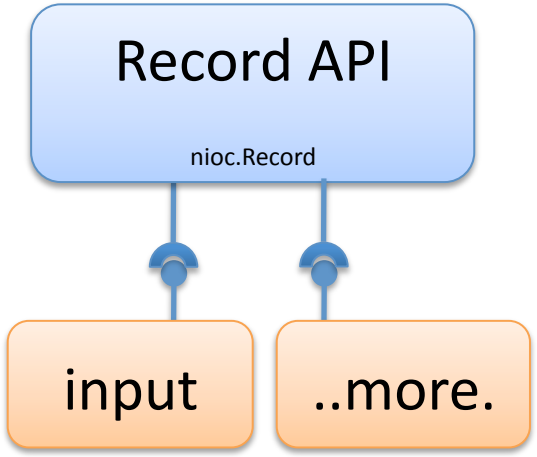
- Link.java
  - API for a read/write link
- ConstantLink.java
  - Implementation for constant value

```
<input type="const">42</input>
```

- Record.java
  - API for a record
- InputRecord.java
  - Implementation that reads “input” link

```
<record type="input" name="rampup">
  <description>Input example, reads
  <input type="ramp"/>
</record>
```





nioc

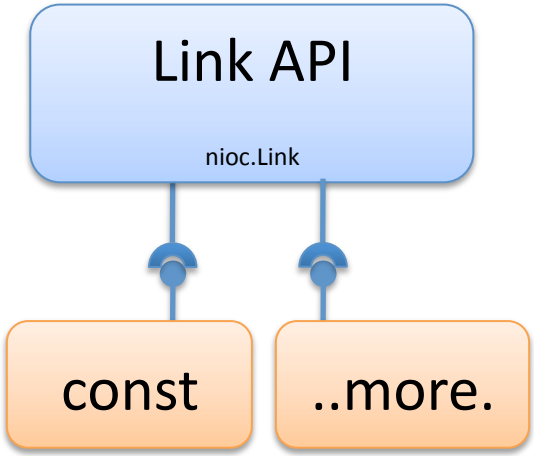
Extension Points

- "Schema" allows plugins to provide implementations
- Registry allows lookup:

  "const" -> nioc.ConstantLink

  "calc" -> nioc.CalcRecord

- Addition of new Link and Record types without recompilation of code that uses them



Creating an extension point can be copy/paste/trial/error.

Google for help.
http://www.vogella.de/articles/EclipseExtensionPoint/article.html

Read Eclipse books.

nioc, nioc.links

Console

Part of Equinox runtime

- **Local access:**

  -console

- **Telnet access:**

  -console 4812

- **Start/stop/ load/unload plug-ins**

- **Add your own commands**

```
osgi> help
---Controlling the OSGi framework---
        launch - start the OSGi Framework
        shutdown - shutdown the OSGi Framework
        close - shutdown and exit
        exit - exit immediately (System.exit)
        init - uninstall all bundles
        setprop <key>=<value> - set the OSGi property
---Controlling Bundles---
        install - install and optionally start bundle from the given URL
        uninstall - uninstall the specified bundle(s)
        start - start the specified bundle(s)
        stop - stop the specified bundle(s)
        refresh - refresh the packages of the specified bundles
        update - update the specified bundle(s)
---Displaying Status---
        status [-s [<comma separated list of bundle states>]  [<segment of bsn>]] - display installed bundles and register
        ss [-s [<comma separated list of bundle states>]  [<segment of bsn>]] - display installed bundles (short status)
        services [filter] - display registered service details
```

...

```
        unschedApp <application id> - unschedules all scheduled applications with the specified application ID
---NIOC commands---
        dbl     - List all records and their value
        records - List available Record types
        links   - List available Link types
        trace   - Toggle processing trace
```

nioc ConsoleCommands.java

Running NIOC Example

- Menu Run Configurations, New OSGi Framework
  - Select nioc.* plugins and required Bundles
- Console Commands
  - help
  - trace, dbl, records, links
  - ss
  - stop, start
  - refresh - after for example small change in CalcRecord.process()
  - close

Adding a new Link

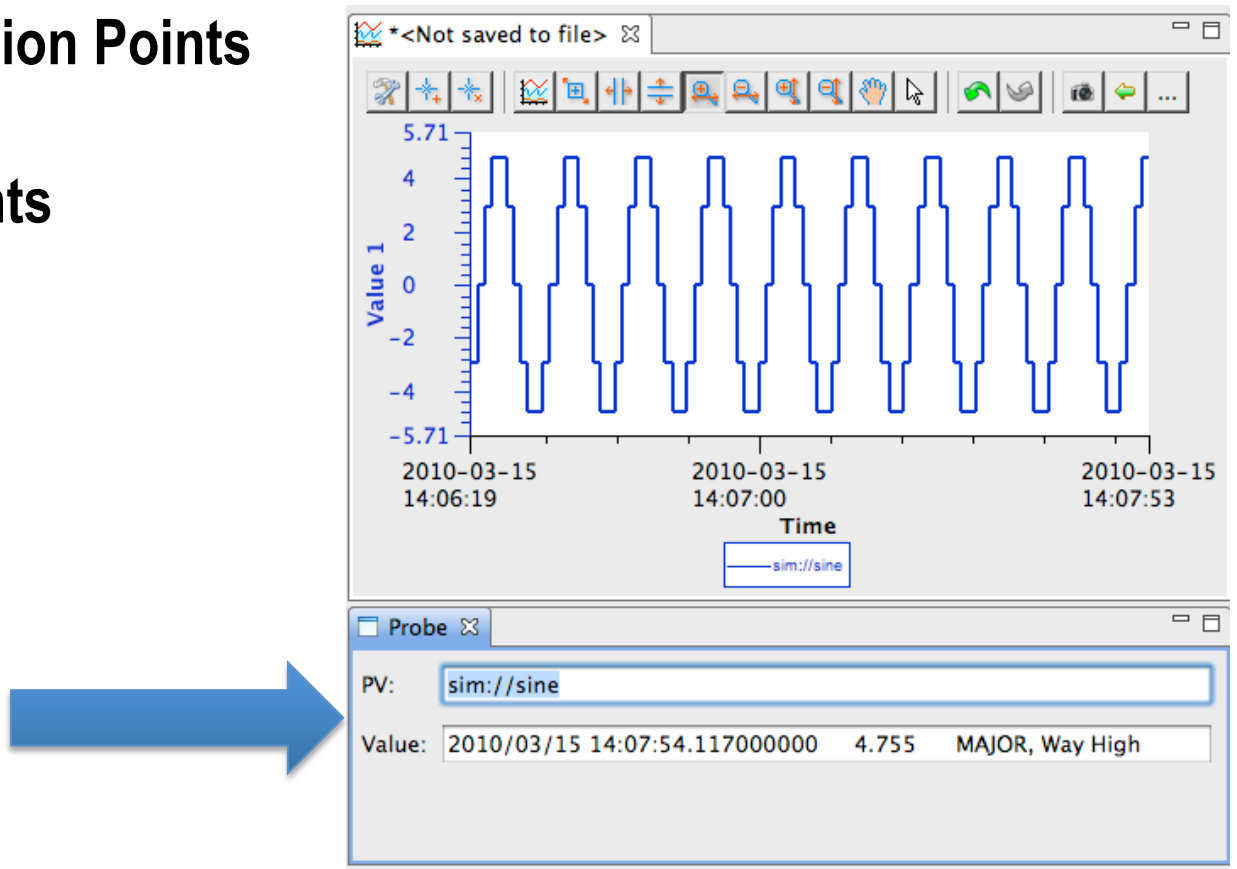
- Add link type “one” that always returns the value 1.0
  - New or existing plug-in
  - Dependency: “nioc”
  - Extensions: Add “nioc.link”
    - type: “one”
    - class: “nioc.link.ExampleOneLink”
  - Implement the class, as simple as

    getValue() { return new Double(1.0); }

“Probe” Project

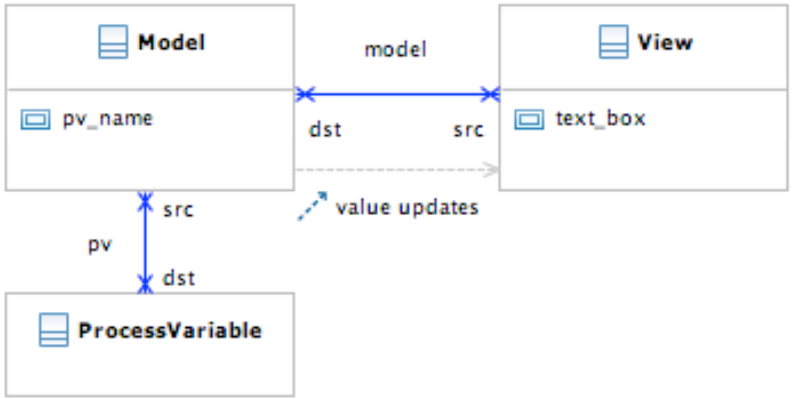
- GUI Tool for CSS
  - Enter PV name, see values

- RCP/CSS Applications are all about Extension Points
  - GUI: Eclipse Extension Points
  - Channel Access:
    CSS Extension Points



Probe Project Setup

- Create new ‘project’
  - Use ‘plugin’
  - Call it ‘yetanotherprobe’
  - Add dependencies
    - org.junit4 – For JUnit tests
    - org.csstudio.utility.pv, …pv.simu – To access PVs
- Think about ‘Model’
  - Takes PV name, connects to control system PV
  - Sends update when PV changes



Again: Test-Driven Development

Start(!) with the 'ModelTest'

- API evolves as you're actually trying to use it:

```
// Create Model
model = new Model();
model.setPVName(…);
// Somehow register for updates
…
```

- Eclipse helps to create code
- Refactor, navigate code, …
- When test compiles, implement Model until it actually 'works'

yetanotherprobe0, 1

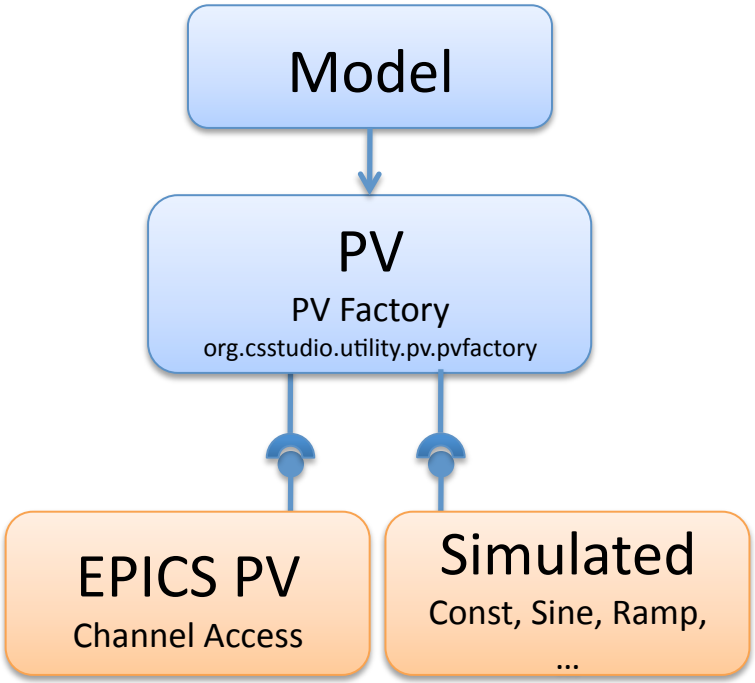
Using A CSS Plug-in

- Simulated PV: org.csstudio.utility.pv.simu
  - SimulatedPVFactory creates simulated PVs
  - Plug-in Dependencies handle the CLASSPATH

yetanotherprobe2

CSS Extension Points

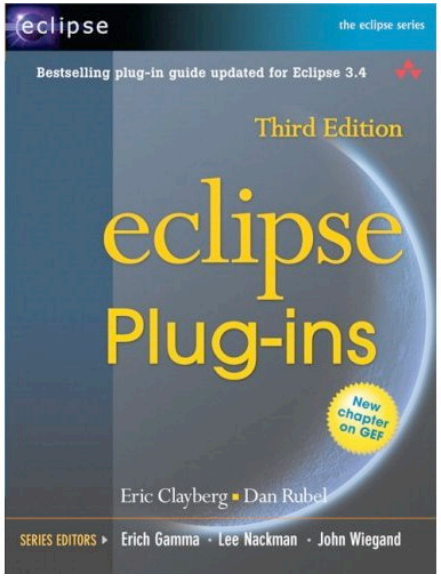
- Idea: Application uses generic API
  - PV, Archive, Logbook, Authentication, …
- Pluggable implementations
  - EPICS, RDB, SNS ELog, LDAP, …
- Extension Points, example PV
  - PV plugin defines extension point:
    PVs for what prefix do you provide?
  - EPICS PV: *I understand "ca://…" PVs*
  - Simulated: *I support "const://..", "sine://…", …*
- Plugin Registry
  - Lists all plugins
    - PV Factory locates available implementations
  - Loads them as needed, including dependencies



yetanotherprobe3

Eclipse Rich Client Platform (RCP)

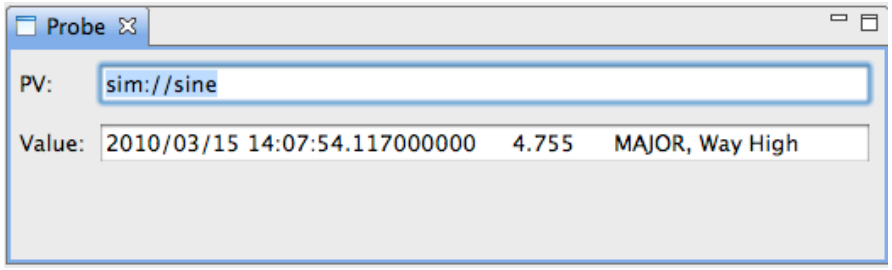
- RCP = Window framework
  - Menu bar, Tool bar, Status bar
  - Multiple document “editors”, “view”
  - Background jobs, file system with change notification, …
- All based on extension points



There is no way to learn about this in one presentation.
Read the books, study online tutorials for a month.

Add GUI to Probe

- Add "View" to MANIFEST.MF, plugin.xml:
  - Dependency: Add org.eclipse.ui
  - Extensions: Add org.eclipse.ui.views
  - id, name, class: Enter something
  - Implement View for the Model
- Then run CSS with the new probe plugin included
  - Where is the new Probe?
    Hollywood Principle, Window/Show View/…



yetanotherprobe4

RCP GUI Notes

- SWT, not AWT
  - ~~Better, worse~~? Different.
- Layouts: Many.
  - GridLayout: Fairly generic and easy to understand
  - FormLayout: More coding, but allows for runtime changes to hide/show widgets
- Beware of GUI thread vs. other control system threads
  - Use Display.asyncExec(Runnable …)

yetanotherprobe4

Exporting a ‘Product’

- Example: ‘Basic EPICS’ CSS.product
  - Add yetanotherprobe plugin to Dependencies
  - Export as “Eclipse Product”

What else is there in Eclipse?

- String Externalization
- Extension points for adding to main menu
  - Entry that would 'open' our Probe view
- Editors: Similar to Views
  - In center of workbench window
  - Have *input*, typically file
  - Examples: SDS, BOY, Data Browser, …
- Preference System, Online Help
- Object Contributions
  - Allow 'Probe' to appear in context menus of other applications
  - .. when a Process Variable is selected

What else is there in CSS?

- Suggested name and hierarchy for plugin names, menu, preferences, online help
- Types for Object Contributions, Drag & Drop
  - IProcessVariable, IFrontEndControllerName, …
- Plugins for
  - Logging
  - Authentication/Authorization
  - …

Summary

- Eclipse IDE
  - Supports test-driven development
- Plug-ins
  - Simplify handling of Dependencies
- Extension Points, Registry
  - Decouple APIs from Implementations
- RCP, CSS
  - A whole software ecosystem built on Extension Points

Steep Learning Curve for Developers

- … but also many Books, online Tutorials
- Invest to learn it, and you'll like it

Results in good product for end users

Image Source:
ORNL General Employee Training Web