Simulate This! - The Eclipse Platform as an Integrated Computational Environment

Jay Jay Billings
Research Staff,
Oak Ridge National Laboratory
billingsjj@ornl.gov
@jayjaybillings
+jayjaybillings
20130327
Outline

• Shell vs. Graphical system
• About the project (NiCE)
• Eclipse-based dev environment & Tycho build
• Adopting Eclipse
• RCP & RAP
• 3D graphics in Eclipse
• Cross-bundle Persistence with Gemini
• Interactive data views with SWT & HDF5
• Examples

Additional Resources:
Go to https://niceproject.sourceforge.net
See “EclipseCon 2013 Presentation page
And with no further ado...
What's the problem with my shell!? 

ORNL's simulation suites run on private clusters, “small” (1000 core) and “leadership class” (see sidebar) machines

We have a broad spectrum of potential users
• “Internal” R&D activities
• Other R&D areas (e.g. Advanced Reactors)
• Technology designers (e.g. Westinghouse, GE-Hitachi)
• Regulators (e.g. NRC)
• Utilities and operators (e.g. TVA)

Only a small percentage of our potential users know what Linux, SSH, MPI and other “common” elements of our computational environment are. They are not experts!
Modeling and Simulation Made NiCE

The NEAMS Integrated Computational Environment (NiCE) helps with all of those difficult chores for the non-expert users…

- Serial, chained, simultaneous or stand-alone job launch and monitoring
- Managing inputs, geometry, materials, and meshing
- Data analysis and visualization
- “Asset” management in multiple formats (SQL, XML)
- Linux, Windows, Mac – soon Web and Android
- Adding new launchers to NiCE (yes, it can modify itself!)

100% Free and Open Source, Eclipse-based
https://niceproject.sourceforge.net
Fun Code Facts

- 100% open source and based on the Eclipse Platform
- Developed at Oak Ridge National Laboratory
- 9 team members, most part-time, 7 contributors + 2 support staff
- Roughly 80k lines of code
  - 49%-51% comments (*very well commented!*)
  - 68% Java, 17% C++ (for native API)
  - Includes JUnit tests that run hourly (Hudson)
- Estimated cost (with COCOMO model) ~ $1M and 20 person-years of effort
- Lots of out-of-source documentation (c.f. Project page)
  - Wiki
  - UML Model
  - Javadoc & Doxygen
- Working closely with several large US government programs and others on everything from nuclear energy to astrophysics.
Eclipse-based Development Environment

Everything shown built with Eclipse
- IBM Rational Software Architect for UML Modeling and most development
- Eclipse Tycho used for builds – best way to build your plug-ins!
  - Much easier than PDEBuild
  - Currently requires “2-stage” build because of repo structure
  - Configures and pulls down dependencies
  - Configures and starts JUnit plugin tests
- Heavily dependent on JDT, PDE, CDT, SVN plugins
- Centralized (via VNC) and local dev environment installs
NiCE's Logical Architecture at a Glance

Component Realizations

- **Client** = Eclipse Workbench & JFace
- **Core** = Equinox + JAXRS + Jetty
- **Data Structures** = Plain OSGi bundle
- **Item** = OSGi Declarative Services (Pluggable)

Not shown: I/O bundle for JPA persistence with Eclipse Gemini
**“The Basics”**

NiCE is all about creating things! Click file and choose “Create an Item” or click the button shown at the left.

Items that you create can be manipulated in the “Item Viewer”

Geometries are created with the ShapeTreeView and shapes are manipulated in the transformation view.

Items are processed and their status monitored in the “Form Header” for each Item.

**DenovoMP Launcher 2**  Ready to process!

The DenovoMP is a fuel performance simulator from ORNL which is apart of the Denovo package.
Adopting Eclipse

After picking Java, choosing the Eclipse Platform was a simple decision: What a platform!

- Large body of Eclipse Foundation and third party bundles (very rich ecosystem)
- Large, active community
- Easily extended dynamic framework (OSGi & Equinox)
- Very good integration that we get for free
- *Could not be easily reproduced if we started from scratch!*
Adopting Eclipse: Migrating to Eclipse 4.x

We're migrating to Eclipse 4!
- CSS support will help us with stakeholder-requested customization
- Dependency injection will help with service discovery
- Dependency injection will greatly simplify our code base
- 4.2.2 performance seems faster than 3.7.2

Unfortunately Eclipse Forms are not yet supported! A fully native migration is not possible!

Solution: 3.x to 4.x bridge!
- Requires 4.x target and ~6 lines in plugin.xml
- Works perfect and quick to do. Target took ~15min, plugin.xml took two.
**Important Note:** We count the *Eclipse Resources bundle* as part of this!

NiCE has RCP and RAP (prototype) builds

- Sponsors are increasing asking for a “cloud” presence
- RAP offers simple access & institutional installs, RCP offers desktop performance
  - Prototype JAX-RS interface to connect RCP NiCE to RAP NiCE
- Some RAP problems for us
  - Single-sourced build of RCP and RAP is not easy with Tycho
  - Security
  - 3D graphics are not supported
- Makes use of many other plug-ins (i.e. - JSch)
Most HPC simulations require geometry and mesh information
- Often very simple (constructive), hierarchical
- Very obscure formats
- Almost no eye ball tests

**JmonkeyEngine is very cool!**
- 100% Java 3D game engine
- Easy Object-Oriented Design
- Available under 3-clause BSD from jmonkeyengine.org
- Includes far more than just graphics!
- Can break on more recent Linux builds

**And also not very Eclipse friendly...**
- No OSGi bundles (must set bundle dependency)
- Let it unpack its own native libraries!
- No native SWT support – must use SWT-AWT bridge
Cross-bundle Persistence with Eclipse Gemini

Persisting classes defined across multiple bundles is not supported in Gemini/EclipseLink, so we cheat and use two databases:

Only works because of original non-JPA design!

Other big challenge with Gemini: bundle start levels! (check NiCE wiki & Eclipse forums for correct levels)
**Full Service Nuclear Data Management**

Nuclear reactor simulators all write to different output formats, making comparative analysis nearly impossible. Ideally it would be very simple:

So we built a system to do it...

- Hierarchical Data Format back-end file
- Java, C++ bindings
- C and F90 bindings soon
- “Automatic” viz via new nuclear reactor-specific SWT based OSGi service
Nuclear Reactor Views with SWT & HDF5

Step 1: Load and select data from an Hierarchical Data Format file (HDF5)

Step 2: Check out cool pin map for obvious problems

Step 3: Look at “power map” of fuels pins

Step 4: Graph pin powers against each other

SWT works exceptionally well with complicated, interactive, scientific views of data!

Adding anomaly detection and direct reference comparisons Soon!
Examples
Domain Plug-ins “On Deck” or not pictured

- Additional nuclear reactor and fuel simulators (about 7)
- Input setup, job launching and analysis for batteries, computational fluid dynamics and tokamak simulators
- *All due by Sept. 30th 2013!*

**Important Note:** These screenshots all look very similar – that's the point! :-)

Adiabatic Quantum Computing

Configuring a binary optimization problem for simulation in a virtual adiabatic quantum computer
Launching the adiabatic quantum computer simulator
Simulating Nuclear Reactors

Configuring a nuclear reactor simulator with the Eclipse 4 build of NiCE
Simulating Nuclear Fuel

Configuring the input for a nuclear fuel simulation in the Eclipse 4 build of NiCE
Mesh-based output data can be statically rendered in NiCE (e.g. - with VisIt from LLNL)
Launching Multiple Jobs (Simultaneously)

Setting up 10 jobs to be launched at once with Eclipse 4 build of NiCE
Closer look at that assembly...

17x17 PWR Assembly in NiCE's geometry editor
Please Give Feedback on the Sessions!

1. Sign In: www.eclipsecon.org

2. Select Session Evaluate

3. Vote +1 0 -1

Making ALM Work - Transform your Application Lifecycle Management to Foster Innovation (presented by HP)

Ronit [HP]