

---

# National Energy Research Scientific Computing Center (NERSC)

## Eclipse as Physicist Work Environment

Wim T.L.P. Lavrijsen, Sebastien Binet  
NERSC HENPC, LBNL  
CHEP – Mumbai, February 2006



# What is Eclipse?

- ***Eclipse is:***

an open source community whose projects are focused on providing a vendor-neutral open development platform and application frameworks for building software.

- ***The Eclipse Foundation is:***

a not-for-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services.

- ***Variety (!) of big-name players:***

– IBM, Intel, Borland, CA, BEA, Sybase, Nokia, Wind River, HP, SAP



# Best known face of Eclipse: IDE

The screenshot displays the Eclipse IDE interface with several key components highlighted:

- Code editor:** The central workspace shows a Python file named `*BasicShell.py` with code for logging configuration, shell prompts, and a `waitForPattern` function. A yellow box labeled "Code editor" points to this area.
- PyLint:** A yellow box labeled "PyLint" is positioned over the code editor, indicating the static analysis tool's integration.
- Code completion:** A yellow box labeled "Code completion" points to a popup menu showing a list of system constants (e.g., `POLLHUP`, `POLLIN`, `POLLMSG`) as the user types `select` in the code.
- Module content:** A yellow box labeled "Module content" points to the right-hand pane, which displays the class and method structure of the `BasicShell` module, including `__doc__` and `__string__`.
- CVS tree:** A yellow box labeled "CVS tree" points to the left-hand pane, which shows the project's file structure under the `>AthASK` directory.
- Interactive prompt:** A yellow box labeled "Interactive prompt" points to the bottom console window, which shows the output of the program, including site information and package search results.



## Looks good ... usable?

- **Need recent hardware (OS/JVM dep.)**
  - Ongoing dev. to improve performance
  - Eclipse itself is nicely responsive
    - Editors, wizards, code-completion, etc.
  - Code compilation slower than shell
  - Memory hungry (as is Atlas Software)
    - E.g. can't index/build ROOT on a 4GB box
- **Redundancy: yet another Java ...**
  - GUI toolkit (SWT)
  - Component model (Equinox)

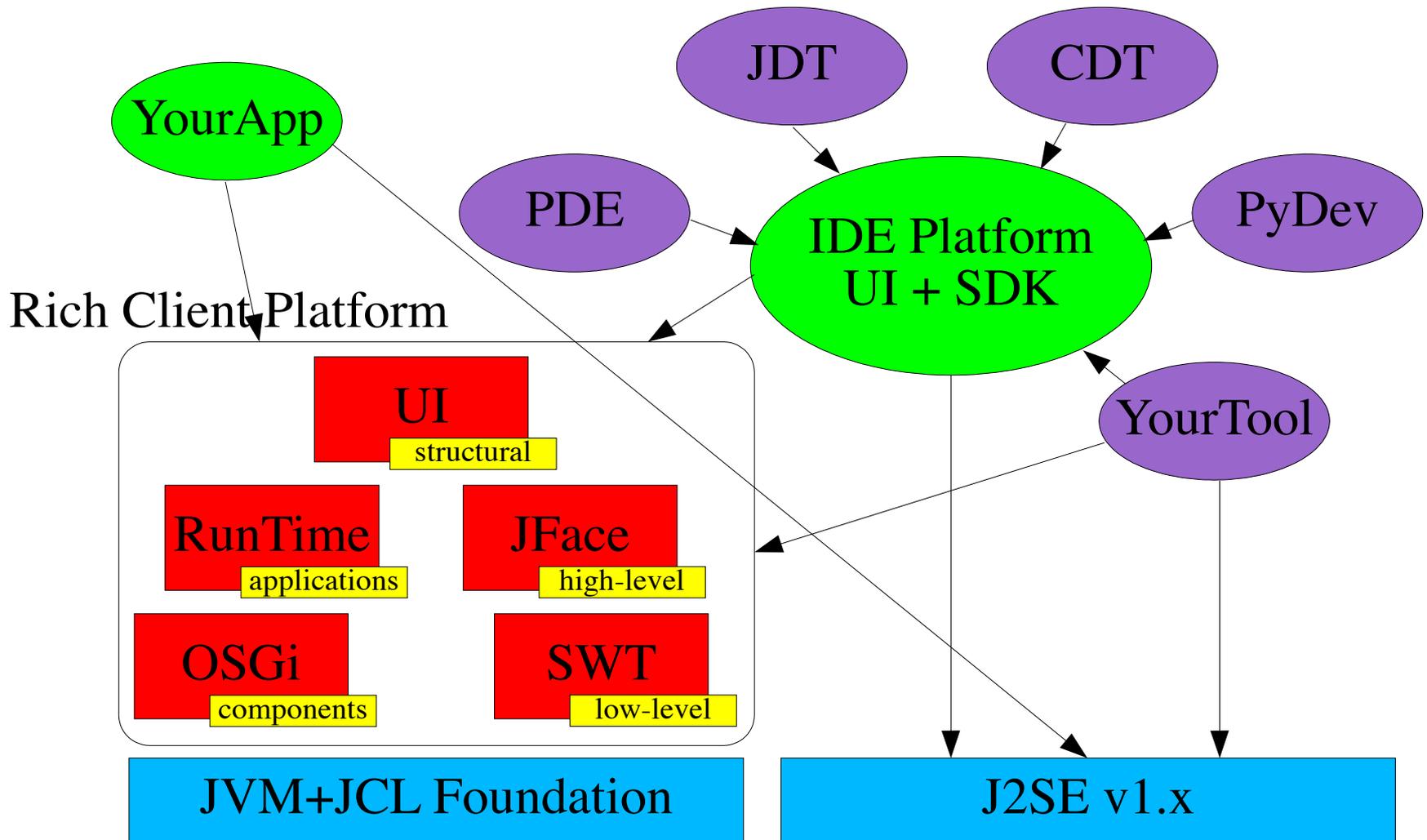


# Atlas problems to solve

- **Software life cycle**
  - Not part of average physicist's toolkit
  - Neither intuitive, nor explorable
- **Tool integration**
  - Multi-language environment
  - Individual tool interactions
  - Setup/environment difficulties
- **Beginners sandbox**
  - More sophisticated “undo” than “rm -rf”



# IDE is just one application ...





# Middleware

- **Not a GUI, but an *application platform***
  - Tools integration
    - Multi-language, -platform, and -vendor
    - Adaptation, distribution (install/updates)
- **Component model implements OSGi**
  - <http://osgi.org> (version 4)
  - Plugins/Bundles w/ life- and runtime mgmt
    - Only connect through extension points
    - Each their own “execution space” (i.e. class loader and process environment)

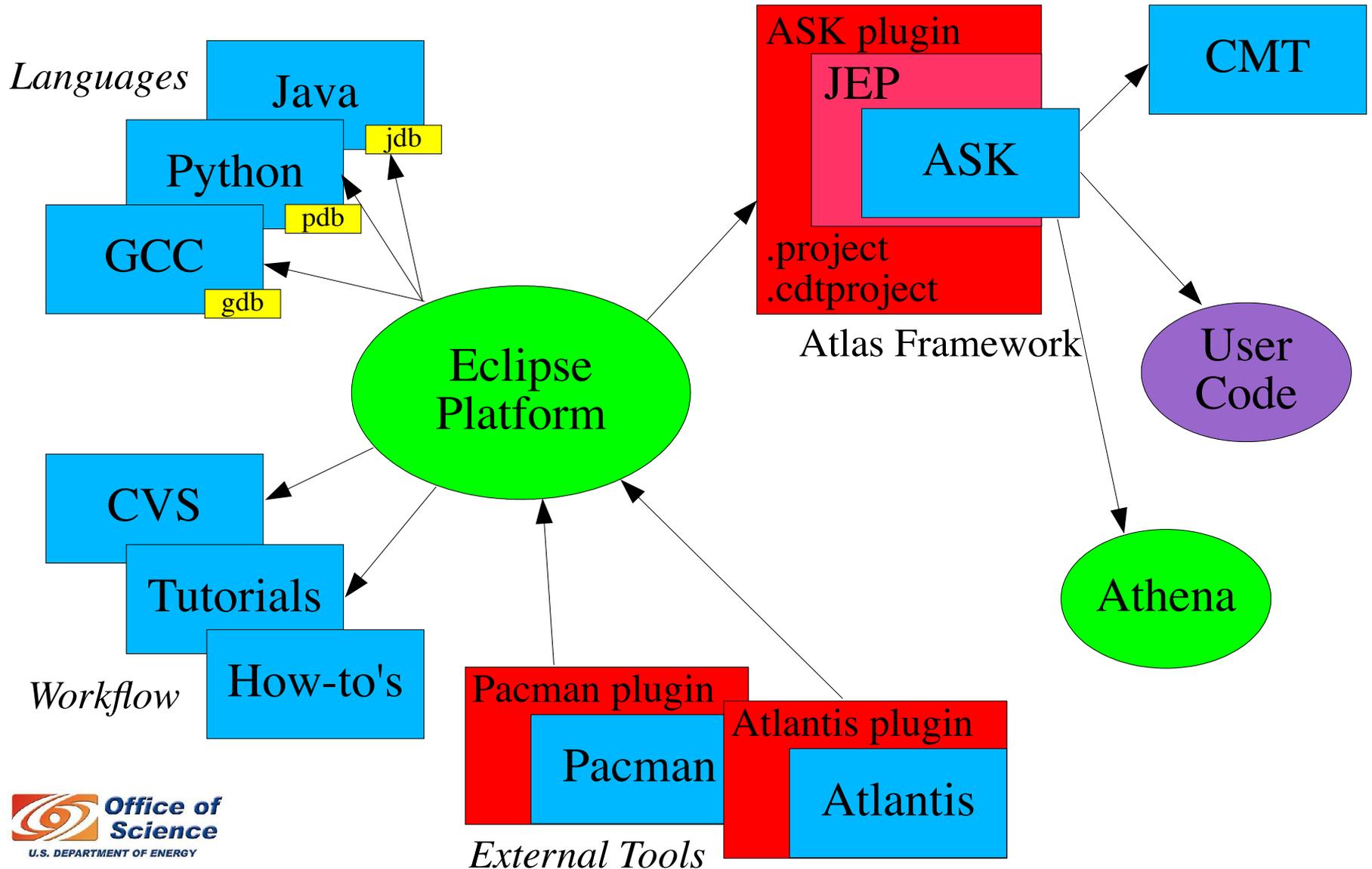


# Open Services Gateway initiative Component Model

- **Complexity solvable by modular sw**
  - *Only* possible if dependencies manageable
    - 100% automatic  $\Leftrightarrow$  100% strict (no backdoors)
    - Limited human input ok, if always verified
  - Shared libs, .class files  $\neq$  modular
    - More like a grab bag: access is not controlled
- **Lifetime control starts at installation**
  - Eclipse: from Eclipse startup
- **Connection control through broker**
  - Not (yet) implemented in Eclipse



# Tools integration





# Athena Startup Kit (ASK)

- **User-space set of tools**
  - Fills voids left by other applications
- **Code generation for Atlas Framework**
  - Algorithms, AlgTools, DataObjects
- **RunTime management**
  - Site abstractions (CERN, BNL, local, etc.)
  - CMT requirements, config, setup
  - POOL file catalog maintainance
- **Module, fully scriptable, CLI, GUI**



# Example Use Case

## End-user Workflow

*Code creation, build, run, and check-in*



# Wizard for new package

The screenshot shows the Eclipse IDE interface with the 'New CMT Package' wizard open. The wizard is titled 'New CMT Package' and contains the following text: 'This wizard creates a new Atlas-style cmt package.' Below this text are four input fields: 'Name' with the value 'MyPackage', 'Release' with '11.0.4', 'Version' with '00-00-01', and 'Location' with '/home/wlav/Eclipse/target/workspace/Example'. A 'Browse...' button is located to the right of the 'Location' field. At the bottom of the wizard are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The background shows the Eclipse IDE with a menu bar (File, Edit, Refactor, Navigate, Search, Project, Run, Window, Help), a toolbar, and a sidebar with tabs for 'C/C++ Projects' and 'Outline'. The 'Outline' tab is active and displays the message 'An outline is not available.' The bottom of the IDE shows the 'Problems' tab with '0 errors, 0 warnings, 0 infos' and a table with columns 'Description', 'Resource', 'In Folder', and 'Location'.

File Edit Refactor Navigate Search Project Run Window Help

C/C++ Projects »1

C/C++ Outline »2

An outline is not available.

### New CMT Package

This wizard creates a new Atlas-style cmt package.

Name:

Release:

Version:

Location:

< Back Next > Finish Cancel

Problems Console Properties

0 errors, 0 warnings, 0 infos

Description	Resource	In Folder	Location
-------------	----------	-----------	----------



# Build with external tool

The screenshot shows an IDE window with the following components:

- Menu Bar:** File, Edit, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard IDE icons for file operations and execution.
- Project Explorer (Left):** Shows a project structure with folders like Binaries, Archives, Includes, cmt, i686-slc3-gcc323-op, InstallArea, MyPackage, MyPackage-00-00, MyPackage, MyAlg.h, share, src, components, MyPackage\_entries, MyPackage\_load.c, MyAlg.cxx, and run.
- Editor (Center):** Displays the source code for `MyAlg.h` and `MyAlg.cxx`. The code includes a `initialize()` function that logs a message and returns `STATUS_SUCCESS`.
- Outline (Right):** Lists the project's symbols, including `MyPackage/MyAlg.h`, `StoreGate/StoreGa`, `StoreGate/DataHar`, `DataModel/DataVec`, `DataModel/Element`, `EventInfo/EventInf`, `EventInfo/EventID`, `GaudiKernel/MsgSt`, `GaudiKernel/ISvcL`, `MyAlg::MyAlg`, `MyAlg::initialize`, `MyAlg::execute`, and `MyAlg::finalize`.
- Building Workspace Dialog (Overlaid):** A modal dialog box titled "Building Workspace" with a lightbulb icon. It shows "Building all..." with a progress bar and the text "Invoking Command: cmt bro make clean all". It has buttons for "Run in Background", "Cancel", and "Details >>".
- Console (Bottom):** Shows the output of the build process, including the command `(construents.make) starting configclean` and the result `allclean ok.`



# Add to repository

**Share Project**

**Enter Module Name**

Select the name of the module in the CVS repository.

Use project name as module name

Use specified module name:

Use an existing module (this will allow you to browse the modules in the repository)

- binet
  - Aod
  - Bazaar
  - Control
  - External
  - Grid
  - PhysicsAnalysis**
  - Simulation

< Back    Next >    Finish    Cancel

return StatusCode::SUCCESS;

Problems Console Properties

C-Build [Example]

```
-----> (constituents.make) building install_includes.make
Document install_includes
-----> (constituents.make) Starting install_includes
No standard include file area
-----> (constituents.make) install_includes done
```



# Conclusions and Outlook

- **Eclipse has lots of potential uses**
  - Impressive, full-featured IDE
  - Individual tool execution environments
  - Allows interactive how-to's
- **Plugin development is easy and clean**
  - Rewarding for post-doc/student
- **Maybe too top-heavy (esp. GUI)**
  - Atlas sw itself has high requirements