

## Cupid Tutorial

This tutorial will take you through the process of installing the Cupid Eclipse Plugin and using some of its basic functionality including reverse engineering an ESMF/NUOPC codebase and performing automatic code generation.

There are two major phases to the tutorial. In the first, you will set up your computational environment to compile and execute ESMF/NUOPC applications. We assume that you already have knowledge of ESMF and NUOPC. For more information about these, visit the following URLs:

- ESMF Home Page: <http://www.earthsystemmodeling.org/>
- NUOPC Home Page: <http://earthsystemcog.org/projects/nuopc/>

### **Set up your computational environment**

First, you need an environment set up where you can compile and run NUOPC applications. In many cases, you will already have this environment set up. It might be on your local machine, or it might be a remote machine accessible via SSH. The details of setting up the environment are not included here, but the steps are outlined below.

*For those with access to Amazon EC2, an Amazon Machine Image (AMI) can be made available with all or prerequisite software preinstalled. To set up your computational environment on a traditional cluster, follow the steps below.*

1. **Install ESMF version 6.2.0 or later.** ESMF is available for download here:  
<http://www.earthsystemmodeling.org/download/releases.shtml>

Detailed install instructions are here:

[http://www.earthsystemmodeling.org/esmf\\_releases/non\\_public/ESMF\\_6\\_2\\_0/ESMF\\_usrdoc/node6.html](http://www.earthsystemmodeling.org/esmf_releases/non_public/ESMF_6_2_0/ESMF_usrdoc/node6.html)

2. **Download the *AtmOcnLndProto* example NUOPC application.** We will use this code to show the features of Cupid.

A description of the NUOPC prototype codes is here:

[http://earthsystemcog.org/projects/nuopc/proto\\_codes](http://earthsystemcog.org/projects/nuopc/proto_codes)

The AtmOcnLndProto source code can be acquired on SourceForge, either via a svn checkout or by downloading a snapshot.

[http://sourceforge.net/p/esmfcontrib/svn/HEAD/tree/NUOPC/tags/ESMF\\_6\\_2\\_0/AtmOcnLndProto/](http://sourceforge.net/p/esmfcontrib/svn/HEAD/tree/NUOPC/tags/ESMF_6_2_0/AtmOcnLndProto/)

Place the AtmOcnLndProto code into a directory by itself, such as  
`/home/user/AtmOcnLndProto`. Ensure that you can compile and execute the example application. An example session looks like this:

```
$ cd ~/AtmOcnLndProto
$ export ESMFMKFILE=/path/to/your/esmf.mk
$ make
```

...build output...

```
$ mpirun -np 4 ./esmApp
```

...execution output...

Examine the PETX.ESMF\_LogFiles to ensure that there were no errors during the run.

### **Set up your Eclipse environment**

#### **1. Download and install Eclipse for Parallel Application Developers, version 4.3.1 (Kepler).**

The main download page is: <http://www.eclipse.org/downloads/>.

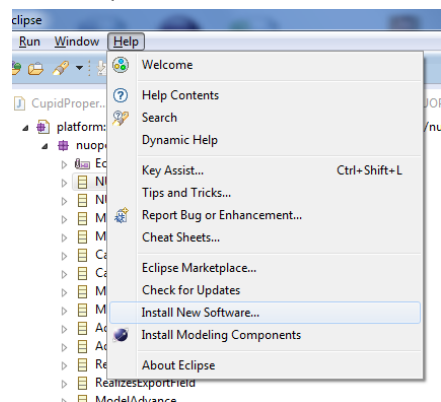
There is a list of available Eclipse packages. Be sure to choose “Eclipse for Parallel Application Developers” as it will come pre-bundled with the necessary plugins for working with remote systems.

#### **2. Unpack the downloaded file into a local directory and run Eclipse by double clicking on the Eclipse executable.**

The first time you start Eclipse, you will be prompted to select a location for your workspace. Choose an empty folder.

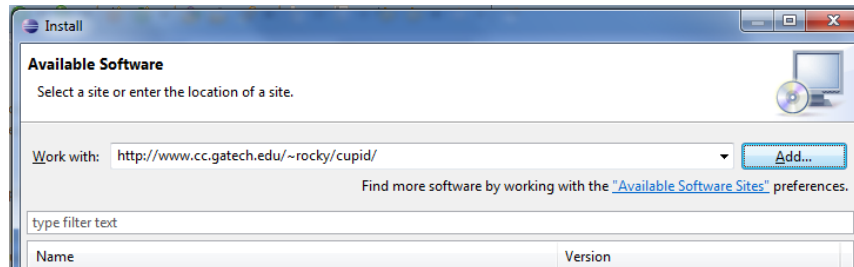
#### **3. Install the Cupid Plugin from the Cupid Update Site.**

##### **a. Click Help→Install New Software**

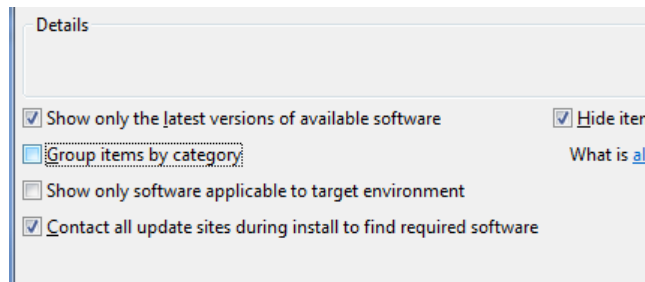


- b. Put the Cupid Update Site URL into “Work with...” You will be prompted to give the update site a name of your choosing.

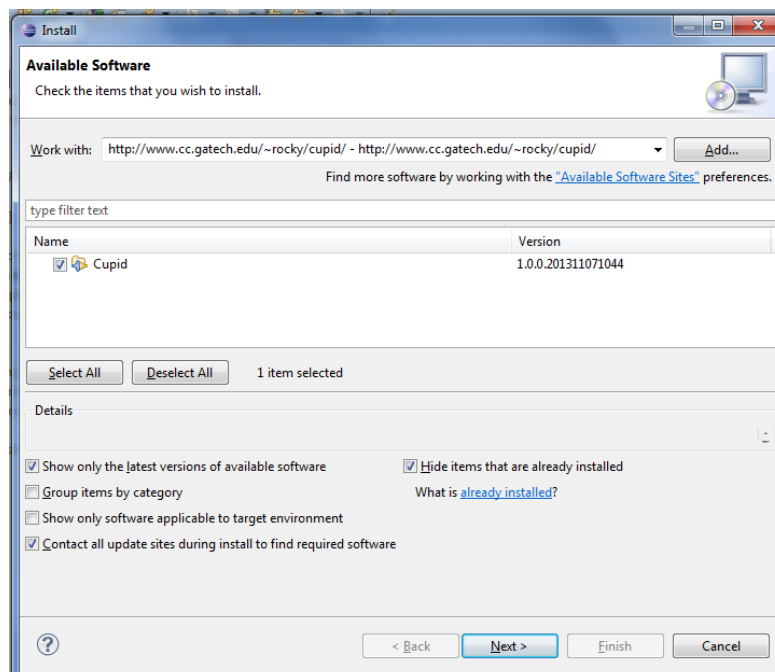
The URL is: **<http://www.cc.gatech.edu/~rocky/cupid/>**



- c. Uncheck the “Group items by category” option.

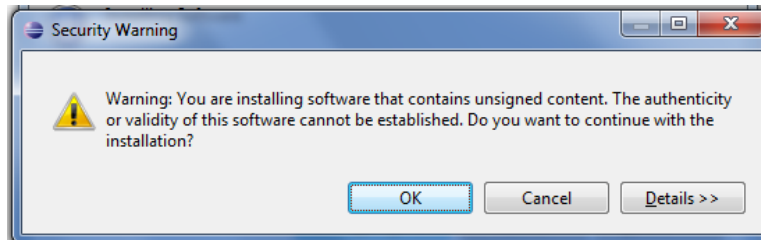


- d. Select “Cupid” from the list and click Next.



- e. You will need to click Next a couple more times and accept the license agreements. Then click Finish. The Cupid plugin and its dependencies will be downloaded and installed.

During the process, you may receive a message that the software contains unsigned content. Click OK.

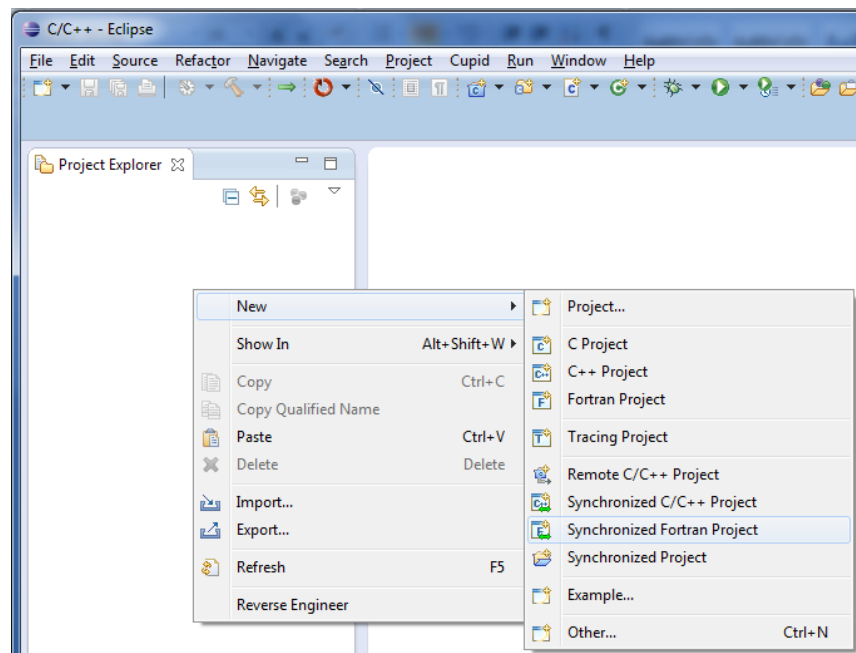


- f. After installation, you will be prompted to restart Eclipse. Click Yes.

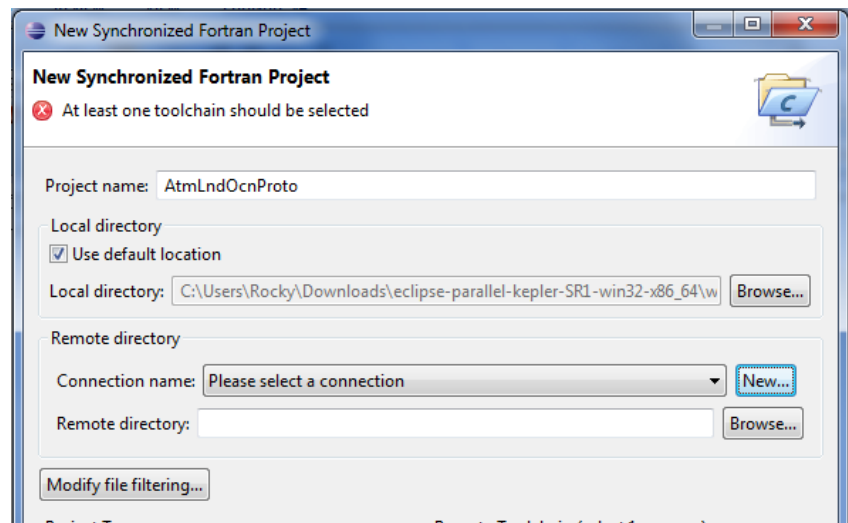
#### 4. Create a new *Synchronized Fortran Project* linked to the NUOPC AtmOcnLndProto source on your computational environment.

Here we assume that you are compiling and executing code on a different machine than where you have installed Eclipse. From now on we will refer to the computational environment you set up as the remote machine.

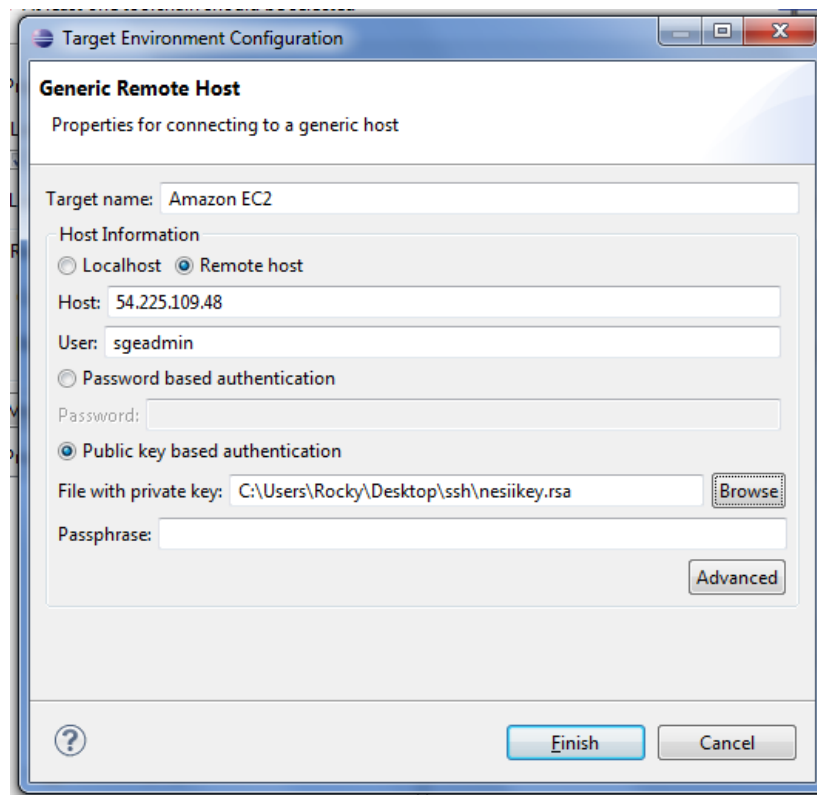
- a. Right click on the Project Explorer and choose New → Synchronized Fortran Project.



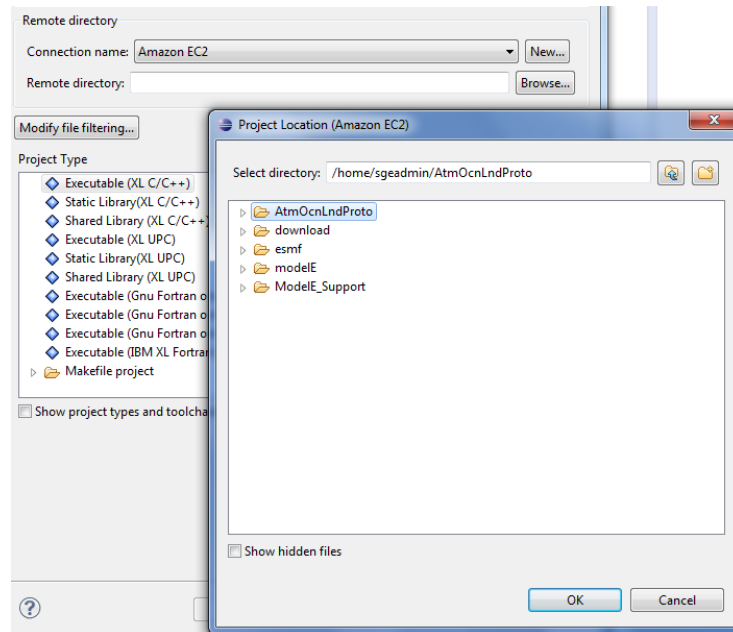
- b. Fill in the project name with “AtmOcnLndProto”.



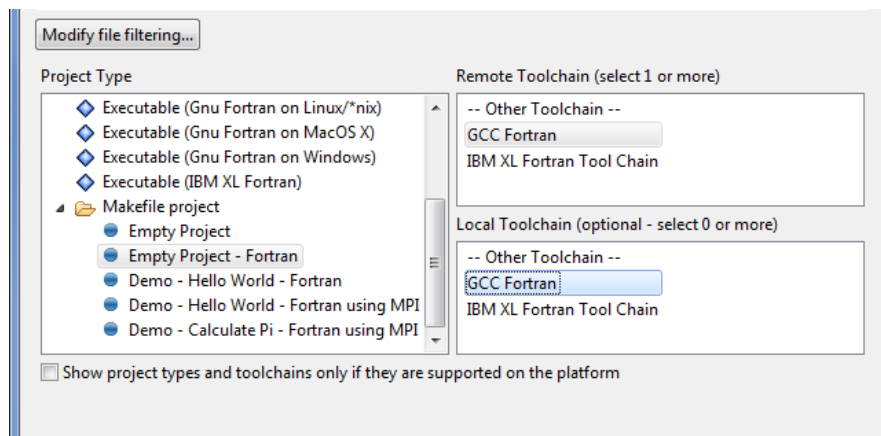
- c. Under Remote directory, click New to create a new connection. Fill in the connection information required to access the remote machine where your NUOPC code is located. Click Finish.



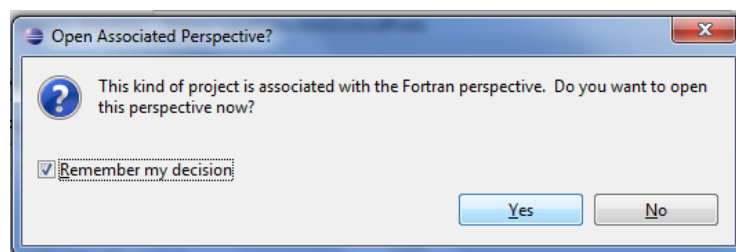
- d. Click Browse and select the remote directory containing the AtmOcnLndProto code.



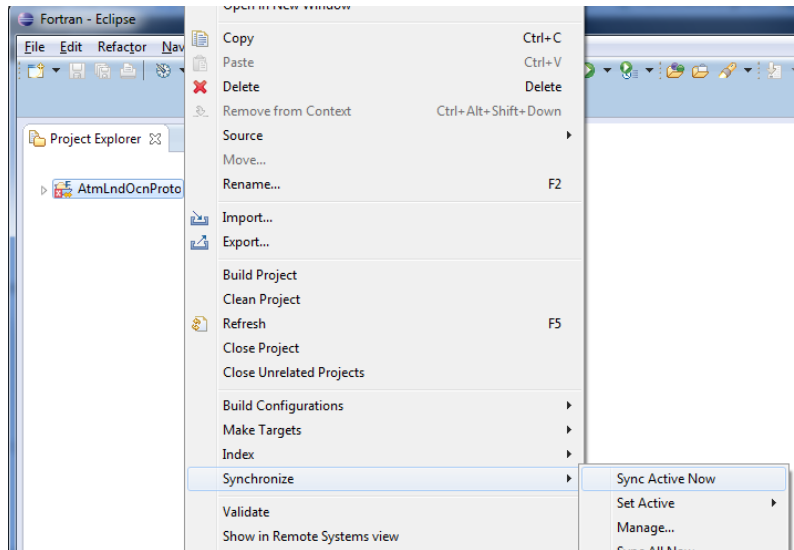
- e. Under Project Type, choose Makefile project → Empty Project - Fortran.  
f. For the Remote and Local Toolchains, choose GCC Fortran.



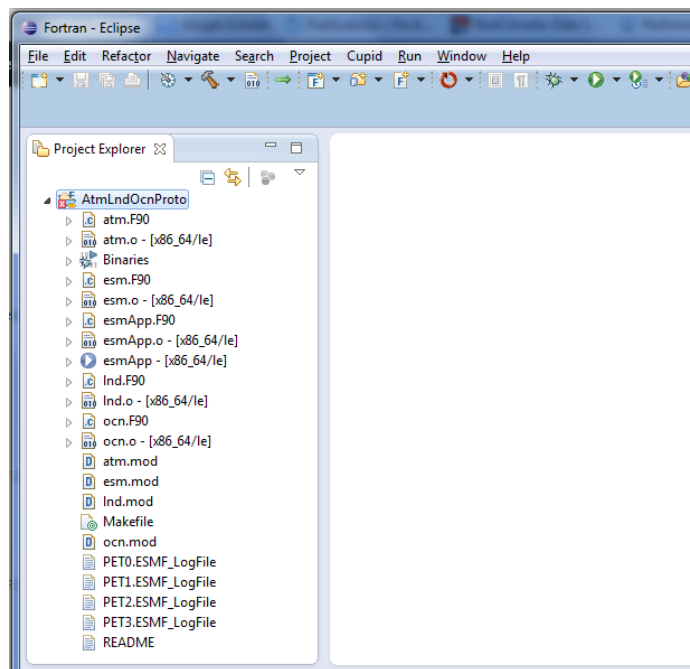
- g. Click Finish. You will be asked if you want to switch to the Fortran perspective. Click Yes.



- h. You will now have a new (currently empty) Fortran project that will be synchronized with the code on the remote machine. To synchronize, right click on the AtmOcnLndProto project folder in the Project Explorer, and choose Synchronize→Sync Active Now.



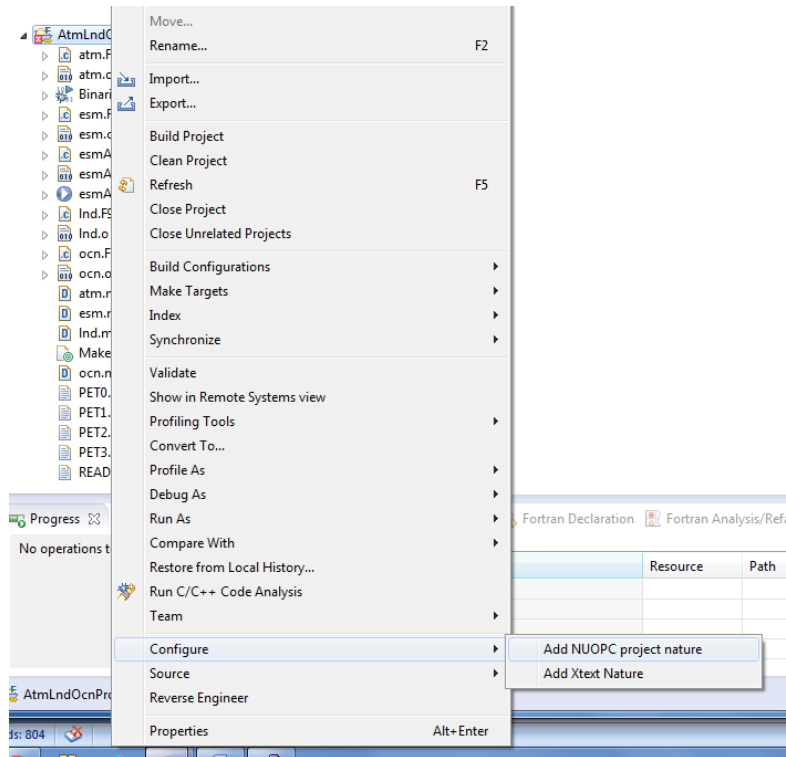
- i. After the synchronization process, you should have a copy of the remote files on your local machine. Eclipse will attempt to keep the files synchronized such that changes made locally are automatically propagated to the remote machine. You can always force a resync by doing the step above.



## 5. Add the NUOPC nature to the AtmOcnLndProto project.

The following step is necessary to let the Cupid plugin know that the AtmOcnLndProto project is a NUOPC-compliant codebase.

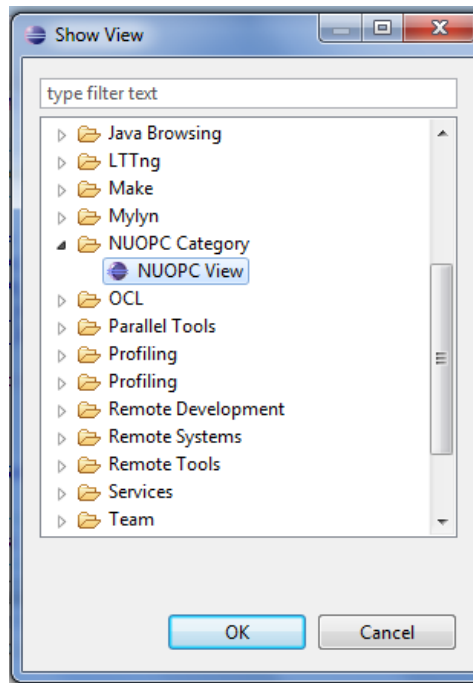
Right click on the AtmOcnLndProto project and choose Configure→Add NUOPC project nature. This step is only required once for new projects. ***Cupid will NOT function without first adding the NUOPC project nature.***



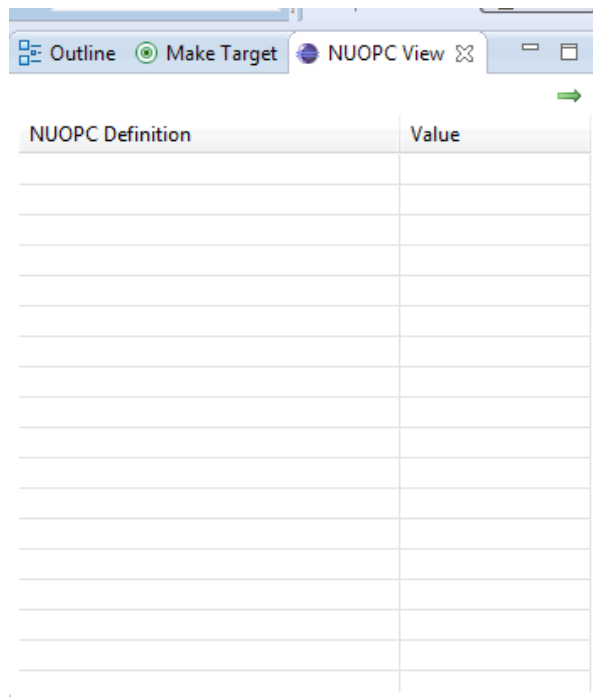
## 6. Show the NUOPC tree viewer.

Click the Window menu→Show View→Other and then select the NUOPC View in the NUOPC Category.





The NUOPC view will appear and will initially be empty. The view can be moved by dragging the “NUOPC View” tab to another part of the screen.



## 7. Reverse engineer the AtmOcnLndProto project.

- Open any of the .F90 files in the AtmOcnLndProto project by double clicking on the file name. The selected file tells Cupid which project to reverse engineer. (It is possible to have multiple NUOPC projects in the same workspace.)
- Click the green arrow in the top-right corner of the NUOPC View or select Cupid→Reverse Engineer from the menu. The NUOPC View will update with a tree structure representing the reverse engineered model.

The screenshot shows the 'NUOPC View' window. The table contains the following data:

NUOPC Definition	Value
NUOPC Application	AtmLndOcnProto
name	AtmLndOcnProto
NUOPC Driver	ESM
NUOPC Model	LND
NUOPC Model	ATM
NUOPC Model	OCN

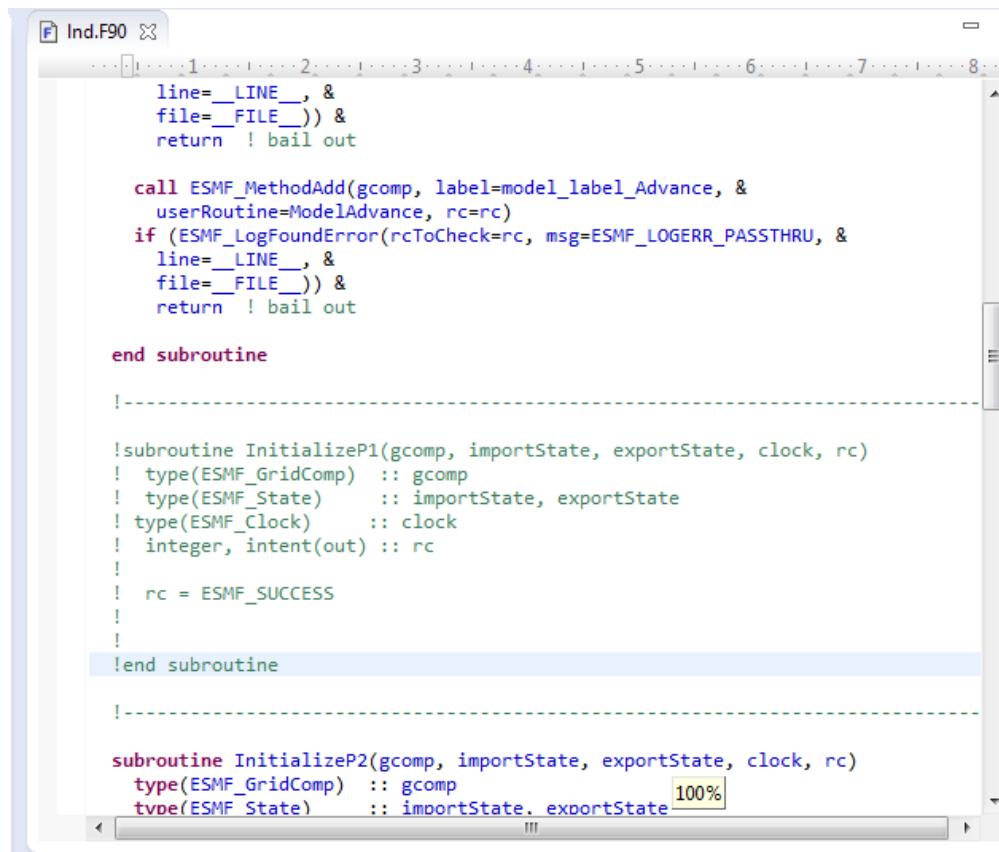
- c. Expand the tree to see what NUOPC code patterns were discovered by Cupid.

NUOPC Definition	Value
NUOPC Application	AtmLndOcnProto
name	AtmLndOcnProto
NUOPC Driver	ESM
NUOPC Model	LND
NUOPC Model	ATM
name	ATM
Imports from NUOPC Model	
Set Services	SetServices
name	SetServices
param_gcomp	gcomp
param_rc	rc
Calls Generic Set Services	
Calls ESMF_GridCompSetEntryPoint	
method	ESMF_METHOD_INITIALIZE
userRoutine	InitializeP1
phase	1
Calls ESMF_GridCompSetEntryPoint	
Model Initialization	
Initialize Phase Definition - IPDv01p1	InitializeP1
Initialize Phase Definition - IPDv01p2	InitializeP2
NUOPC Model	OCN

## 8. Show a NUOPC compliance issue using Cupid.

Cupid can automatically indicate NUOPC compliance problems that would hinder interoperability of a codebase with other NUOPC components. The `AtmOcnLndProto` code is already NUOPC compliant, but we will introduce a problem in order to show how Cupid reports validation errors.

- a. Open the file **Ind.F90** and comment out the subroutine `InitializeP1`. We will assume that the developer forgot to implement this subroutine, or knows about it but simply hasn't written it yet.



```
Ind.F90
1
2
3
4
5
6
7
8
...
line= _LINE_, &
file= _FILE_) &
return ! bail out

call ESMF_MethodAdd(gcomp, label=model_label_Advance, &
userRoutine=ModelAdvance, rc=rc)
if (ESMF_LogFoundError(rcToCheck=rc, msg=ESMF_LOGERR_PASSTHRU, &
line= _LINE_, &
file= _FILE_) &
return ! bail out

end subroutine

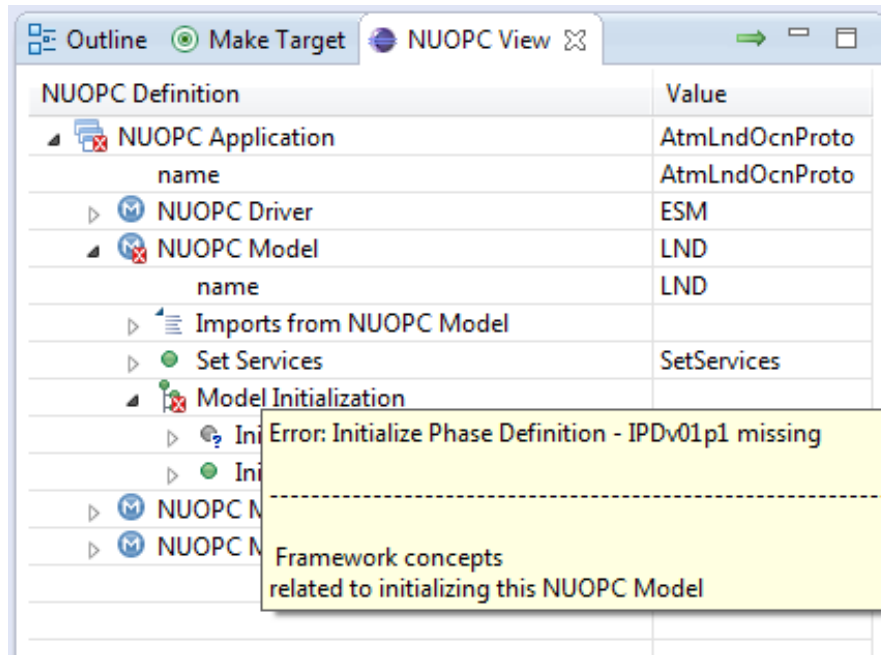
!-----

!subroutine InitializeP1(gcomp, importState, exportState, clock, rc)
! type(ESMF_GridComp) :: gcomp
! type(ESMF_State)    :: importState, exportState
! type(ESMF_Clock)    :: clock
! integer, intent(out) :: rc
!
! rc = ESMF_SUCCESS
!
!end subroutine

!-----

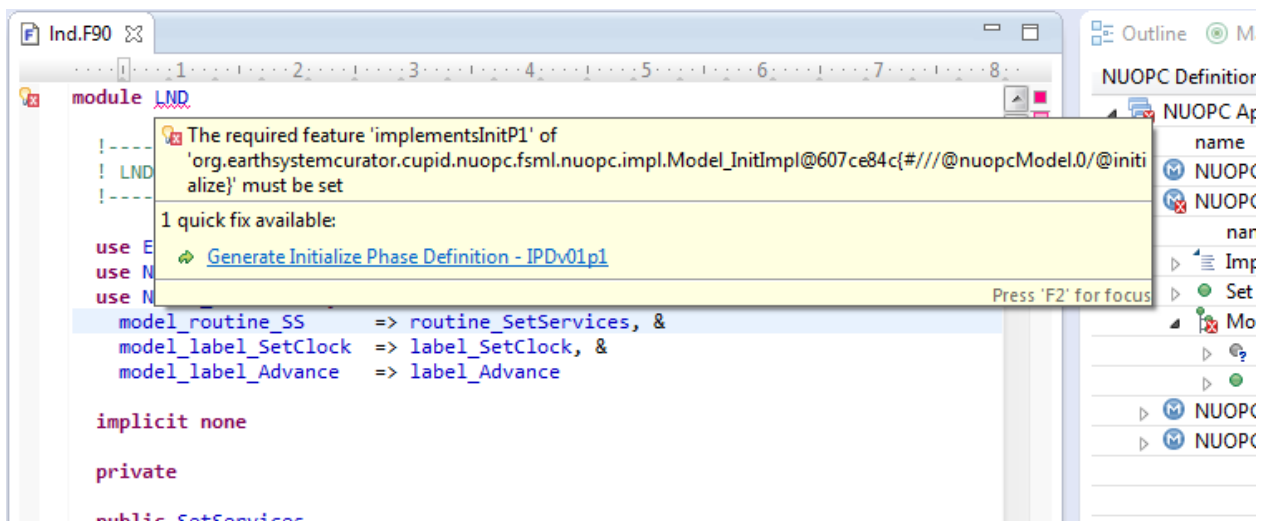
subroutine InitializeP2(gcomp, importState, exportState, clock, rc)
type(ESMF_GridComp) :: gcomp
type(ESMF_State)    :: importState, exportState 100%
```

- b. Save the modified **Ind.F90** file. Then, click the reverse engineer button again to re-analyze the code and update the NUOPC View. The view should now indicate an error with a small red X over the invalid node(s) in the tree. Hovering over the invalid node shows a description of the validation error.



## 9. Use Cupid's Quick Fix feature to generate missing code.

- Scroll to the top of the **Ind.F90** file. The module name **LND** has been underlined in red indicating that the module has failed validation and is not NUOPC compliant.
- Hover over the module name to show available quick fixes.



- c. Click on the one quick fix available: "Generate Initialize Phase Definition - IPDv01p1"
- d. A new subroutine skeleton named InitP1 is generated automatically and inserted at the end of the module. The generated code is bookmarked and highlighted in yellow on the right vertical ruler to make it easy to distinguish the generated code from the existing code.

```

Ind.F90
1: line=__LINE__, &
2: file=__FILE__) &
3: return ! bail out
4:
5: call NUOPC_TimePrint(currTime + timeStep, &
6: "-----> to: ", rc=rc)
7: if (ESMF_LogFoundError(rcToCheck=rc, msg=ESMF_LOGERR_PASSTHRU, &
8: line=__LINE__, &
9: file=__FILE__)) &
10: return ! bail out
11:
12: end subroutine
13:
14: !
15: ! Initialize Phase Definition: IPDv01p1
16: !
17: ! In this phase, models should advertise the set of import and export fields
18: ! This is the list of fields that may potentially be exchanged during coupli
19: ! although some fields may not be connected for a particular configuration.
20: !
21: subroutine InitP1(gcomp, importState, exportState, clock, rc)
22:   type(ESMF_GridComp), intent(inout) :: gcomp
23:   type(ESMF_State), intent(inout) :: importState
24:   type(ESMF_State), intent(inout) :: exportState
25:   type(ESMF_Clock), intent(inout) :: clock
26:   integer, intent(out) :: rc
27: end subroutine
28:
29: end module

```

Cupid generated code

100%