Eclipse UOMo Tutorial

Eclipse UOMo is an API for working with physical quantities and units. We are going to create a demonstration application that provides an API to work with Newton's Second Law of Motion:

"The acceleration of a body is directly proportional to, and in the same direction as, the net force acting on the body, and inversely proportional to its mass."



Otherwise written as: F = M×A Where: F is Force in Newtons M is Mass in Kilograms A is Acceleration in Meters per second Our example API will provide methods for computing any of the above, given the other two.

Example code for the following is available at GitHub:

https://github.com/duckAsteroid/uomo-example

Getting Started - Eclipse Plugin Project...

Most of verbosity in the steps that follows are related to complexities of creating Eclipse plug-ins rather than anything hard about using UOMo...

- 1. In Eclipse click the File -> New -> Project.. menu
- 2. Select Plug-in Project:

🛞 New Project	
Select a wizard	
Create a Plug-in Project	
Wizards:	
type filter text	×
🖄 Java Project	
* Java Project from Existing Ant Buildfile	
🕸 Plug-in Project	
🕨 🗁 General	
Android	
Bndtools]
Eclipse Modeling Framework	
Or Cancella Cancel	el Finish

Click "Next"

3. Enter a name for the plugin project (e.g. com.acme.n2l):

Plug-in Project
Create a new plug-in project
Project name: com.acme.n2l
☑ Use default location
Location: /home/chris/workspaces/units/com.acme.n2l Browse
Project Settings
Create a Java project
Source folder: Src
Output folder: bip
Target Platform This plug-in is targeted to rup with:
Eclipse version: 3.5 or greater 1
O an OSGi framework: Equinox
Working sets: Example \$
(?) < Back Next > Cancel Finish

4. You can now customise the plugin details as follows:

😣 New Plug-in Project		
Content		
Enter the data required to	generate the plug-in.	
Properties		
ID:	com.acme.n2l	
Version:	1.0.0 qualifier	
<u>Version</u>	Nowtops Second Law ADI	
N <u>a</u> me.		
Ven <u>d</u> or:	Acme Inc.	~
Execution Environment:	JavaSE-1.6	‡ Environments
Options		
<u>Generate an activator</u>	; a Java class that controls the plug-i	n's life cycle
Ac <u>t</u> ivator: com.acme.	n2l.Activator	
🗌 T <u>h</u> is plug-in will make	contributions to the UI	
Enable A <u>P</u> I analysis		
Rich Client Application		
Would you like to create a	a rich client application?	○ Yes ● No
?	< Back Next > Cano	el Finish

Ensure that all checkboxes under "Options" are de-selected.

Click "Finish"

If the following dialog appears click "Yes" to open the PDE perspective:



- Eclipse will create your new project. Now we need to configure Eclipse so that it knows about UOMo and where it lives to build against. Click the menu "Window -> Preferences"
- 6. Navigate to the "Plug-in Development -> Target Platform" section of the preferences dialog:

8 Preferences		
type filter text 🛛 🕱	 No target definitirently set active. 	
 General Android Ant Bndtools OSGi Code Recommenders Data Management 	Add, edit and remove target definitions. The definition will be used as the target platfor workspace plug-ins will be compiled and te New definitions are stored locally, but they to a project in the workspace and shared w Target definitions:	ne active target rm which ested against. y can be moved vith others.
 Data Management Help 	Running Platform	Reload
 Install/Update Java 		Add
 Maven 		Edit
▶ Mwe2		Remove
 Plug-in Development API Baselines 		Share
API Errors/Warnings API Use Scans		
Compilers		
Editors		
Target Platform		
 Run/Debug 	Locations:	
▶ Team		
► XML		
 Xtext 	Restore Defaults	Apply
?	Cancel	ОК

Click "Add..."

7. Select "Nothing: Start with an empty target definition" in the "New target definition" dialog:

🐵 New Target Definition				
Target Definition				
Create a new target definition.				\odot
Initialize the target definition with:				
Nothing: Start with an empty target definition				
O Default: Default target for the running platform				
 Current Target: Copy settings from the current target platfor 	m			
○ Template: Base RCP (Binary Only) ‡				
(?)	< Back	lext >	Cancel	Finish

8. Give the new target a name, such as "UOMo":

Target Content Image: Content Edit the name, description, and plug-ins contained in a target. Image: Content Name: UOMo Locations Content Environment Arguments Implicit Dependencies Implicit Dependencies The following list of locations will be used to collect plug-ins for this target definition. Add Edit Remove Update Update Show location content	8 New Target Definition				
Edit the name, description, and plug-ins contained in a target.	Target Content				
Name: UOMo Locations Content Environment Arguments Implicit Dependencies The following list of locations will be used to collect plug-ins for this target definition. Edit Edit Remove Update Update Show location content	Edit the name, description, and plug-ins contained in a target.				\odot
Locations Content Environment Arguments Implicit Dependencies The following list of locations will be used to collect plug-ins for this target definition. Edit Edit Remove Update Update \$how location content	Name: UOMo				
The following list of locations will be used to collect plug-ins for this target definition. Add. Edt Remove Update Update	Locations Content Environment Arguments Implicit Depend	dencies			
Add Edit Update Update	The following list of locations will be used to collect plug-ins for t	his target definition.			
Edit Remove Update					<u>A</u> dd
Remove Update					<u>E</u> dit
Update					<u>R</u> emove
Show location content					Update
Show location content					
Show location content					
□ Show location content					
□ Show location content					
□ Show location content					
□ Show location content					
□ <u>S</u> how location content					
Show location content					
□ <u>S</u> how location content					
□ Show location content					
	□ <u>S</u> how location content				
(?) < Back Next > Cancel Finish	?	<back< td=""><td>Next ></td><td>Cancel</td><td>Finish</td></back<>	Next >	Cancel	Finish

Click "Add ... "

9. Select "Software Site" from the plug-in source options:

8 Add Content	
Add Content	
Select a source of plug-ins.	
🔁 Directory	
💼 Installation	
🖗 Features	
📢 Software Site	
Download plug-ins from a software site such as a p2 repo	ository or update site.
(?) < Back Next >	Cancel Finish

10. Firstly, we need to give our target definition a place to get Eclipse plug-ins from. Enter the URL <u>http://download.eclipse.org/releases/juno</u> into the "Work With" field. Ensure that the "Group by Category" field is un-checked.

Type "Eclipse Platform" into the search field:

😣 Add Content

Add Software Site

Select content from a software site to be added to your target

Work with: http://download.eclipse.org/releases/juno

Add.. V

Work with the list of software sites

Eclipse Platform	()
Name	Version
🗆 🖗 Eclipse Platform	4.2.2.M20130204-1200
🗹 🄯 Eclipse Platform	4.2.1.v20130118-173121-9MF7GHYdG0B5
 Platform Launcher Executables Eclipse Platform Plug-in Developer Resources Eclipse Platform SDK Sapphire Eclipse Platform Support (Incubation) 	3.6.0.v20121119-201001-7P7OG2BFLWUU 4.2.1.v20130118-173121-9MF7GHYdG0B5 4.2.2.M20130204-1200 0.5.4.201302121329
1 item selected Details Common OS-independent base of the Eclipse platform. (Bi	nary runtime and user documentation.)
	Properties
🗌 Group by Category 🛛 🗹 Show only the latest ve	rsion

Included Software

?

By default, all required software is added to the target based on its environment settings. Turning this option off allows software to be added with missing requirements and multiple environments. This setting applies to the entire target definition.

Include all environments ☑ Include source if available ☑ Include configure phase

Select the Eclipse Platform (with out M* at the end :- as this is a milestone release) Click "Finish"

Next >

Cancel

Finish

<Back

11. Eclipse will do some loading/resolving...

Now we need to add the UOMo libraries to our target. Click "Add.." again on the target platform editor window. The select "Software Site" again. But this time enter the URL http://download.eclipse.org/uomo/0.6/repository/

S Add Content	
Add Software Site	
Select content from a software site to be added to your ta	arget
Work with: http://download.eclipse.org/uomo/0.6/repos	sitory/ 🔹 Add
	Work with the list of software sites
type filter text	(*
Name	Version
🔻 🜌 💷 Units of Measurement (UOMo)	
✓ 🏶 Eclipse UOMo	0.6.0
1 item selected	
Eclipse UOMo	
	Properties
Group by Category Show only the latest w	ersion
By default, all required software is added to the target bas this option off allows software to be added with missing re This setting applies to the entire target definition.	sed on its environment settings. Turning equirements and multiple environments.
Include required software	
Include all environments	
Include source if available	
M Include configure phase	
Back N	lext > Cancel Finish

Select the EclipseUOMo feature.

Click "Finish"

12. Eclipse will do some more loading/resolving. You are back at the target platform preferences page:

Ø Preferences		
type filter text 🛛 🕱	Target Platform	> = => - -
 General Android Ant Bndtools OSGi Code Recommenders 	Add, edit and remove target definitions. The definition will be used as the target platfor workspace plug-ins will be compiled and to New definitions are stored locally, but the to a project in the workspace and shared w	ne active target rm which ested against. y can be moved vith others.
 Data Management 	Target definitions:	
▶ Help	🗆 쒭 Running Platform	Reload
 Install/Update Java 	✓ [™] UOMo (Active) - /com.acme.n2l/uor	Add
 Maven 		Edit
Mwe2		Demouro
 Plug-in Development 		Remove
API Baselines		Share
API Errors/Warnings		
API Use Scans		
Compilers		
Editors		
OSGi Frameworks		
Target Platform		
Run/Debug	Locations:	
▶ Team	http://download.eclipse.org/releases/	juno
▶ XML	http://download.eclipse.org/uomo/0.6	/repository/
Xtend	Pestore Defaults	Apply
Xtext	Restore Deradits	- Abbra
?	Cancel	OK

Select UOMo target and click "Apply"

13. (Optional) You can click "Share.." if you want to save this target definition as a file for use in the future. Pick a location and a name for the target file in the window as below:

Share Target Definition
Share Target Definition Choose a workspace location for the target definition file
Enter or select the parent folder:
com.acme.n2l
🔻 d com.acme.n2l
ie → .settings
▶ 🗁 n2l-api
File name: uomo.target
Cancel Finish

Click "Finish" when you are done.

- 14. Click "OK" on the preferences page
- 15. Now you are ready to start work on our N2L plug-in.

Create a new class called NewtonsSecondLaw in the package com.acme.n2l Now add the following code:

```
package com.acme.n2l;
import org.eclipse.uomo.units.SI;
import org.eclipse.uomo.units.impl.quantity.AccelerationAmount;
import org.eclipse.uomo.units.impl.quantity.ForceAmount;
import org.eclipse.uomo.units.impl.quantity.MassAmount;
```

public class NewtonsSecondLaw {

```
public static final ForceAmount calculateForce(MassAmount m, AccelerationAmount a)
{
    double m_kg = m.doubleValue(SI.KILOGRAM);
    double a_si = a.doubleValue(SI.METRES_PER_SQUARE_SECOND);
    return new ForceAmount(m_kg * a_si, SI.NEWTON);
  }
}
```

The important part of this code is the calculateForce method; it takes as parameters an amount of mass and an amount of acceleration - and returns an amount of force. The units of these parameters are not defined; just that they are of quanity Mass and Acceleration respectively. So our code needs to get the absolute value of these in a known unit for calculation - for simplicity we use the SI units Kilogram (kg) and Metres per second per second (m/s²).

We then simply perform the multiplication, and create a result using the SI unit for Force - Newtons (N).

- 16. Now we are ready to test out our API with a Unit test. What follows is a little long winded thanks to how Eclipse Plug-ins do unit tests..
- 17. Firstly we need to create a New Fragment Project (File -> New -> Project...):

🛞 New Project	
Select a wizard	-
Create a Plug-in Fragment Project	
Wizards:	
type filter text	×
🕨 🧀 Java	
🕨 🗁 Maven	
🔻 🗁 Plug-in Development	
👫 Feature Patch	
🎼 Feature Project	
📲 Fragment Project	
Plug-in from Existing JAR Archives	
🕸 Plug-in Project	- 1
♦ Update Site Project	
Or A Back Next > Cancel F	inish

18. Give the fragment the name of our main project with ".test" appended:

8 New Fragment Project
Fragment Project
Create a new fragment project
Project name: com.acme.n2l.tests
✓ Use <u>d</u> efault location
Location: /home/chris/workspaces/units/com.acme.n2l.tests Browse
Project Settings
🗹 Create a <u>J</u> ava project
Source folder: src
O <u>u</u> tput folder: bin
Target Platform This fragment is targeted to run with:
Eclipse version: 3.5 or greater \$
O <u>a</u> n OSGi framework: Equinox ‡
Working sets
✓ Add project to working sets
Working sets: Example \$
? < Back Next > Cancel Finish

Click "Next"

19. We need to give the fragment a "host" that is our com.acme.n2l plugin:

🛞 New Fragment	Proje	ct	
Fragment Content			
Enter the data requ	ired to	o generate the fragment.	
Properties			
<u>I</u> D:		com.acme.n2l.tests	
Version:		1.0.0.qualifier	
N <u>a</u> me:		N2L API Tests	
Vendo <u>r</u> :		Acme Inc	▼
Execution Environr	nent:	JavaSE-1.6	Environments
_		•)(
Host Plug-in			
<u>P</u> lug-in ID:	com.	acme.n2l	Bro <u>w</u> se
<u>M</u> inimum Version:	1.0.0		Inclusive 💲
Ma <u>x</u> imum Version:			Exclusive ‡
?		< Back Next > Cancel	Finish

Click "Finish"

20. Now we need to add JUnit to the fragment's dependencies. Find and double click the META-INF/MANIFEST.MF file to open the fragment manifest editor. Select the "Dependencies" tab:

NewtonsSecondLaw.java	🏽 com.acme.n2l.tests 🛿		•
bependencies			0 🎋 ≉ 🔇
Required Plug-ins	↓ªz	Imported Packages	
Specify the list of plug-ins requi fragment.	red for the operation of this	Specify packages on which this fragment d explicitly identifying their originating plug-i	epends without n.
	Add	org.unitsofmeasurement.quantity	Add
	Remove	🖶 org.unitsormeasurement.unit	Remove
	Up		Properties
	Down		
	Properties		
	Total: 0		
	Total. 0		Total: 2
Automated Management of	Dependencies J ^a z	Dependency Analysis	
verview Dependencies Runtin	ne Extensions Extension Points	Build MANIFEST.MF build.properties	

Click "Add..."

21. Enter "junit" in the search field:

8 Plug-in Selection	
Select a Plug-in	
junit	(*
Matching items:	
🗣 org.junit (4.10.0.v4_10_0_v20120426-0900)	
> org.junit.source (4.10.0.v4_10_0_v2012042	6-0900)
org.junit	
?	Cancel OK
-	

Select the org.junit plugin and click "OK"

- 22. Now we create a unit test class in our fragment. Find and right click on our NewtonsSecondLaw.java file and choose "New -> Other..." from the context menu.
- 23. Select Java -> JUnit -> JUnit Test Case from the dialog:

Select a wizard Create a JUnit Test Case Wizards: type filter text ♥ ☞ Git ♥ ☞ Java @ Annotation @ Class @ Enum	*
Create a JUnit Test Case Wizards: type filter text Git Git Git Git Git Git Git Git Git G	×
Wizards: type filter text Get Git Get Git Git Git Git Git Git Git Git	×
Wizards: type filter text Cut Cut Class Class Cur Enum	×
type filter text ► GIT ► Java ⓒ Annotation ⓒ Class ⓒ Fourm	×
 Git Java Annotation Class Enum 	
 Java Annotation Class Enum 	
Class Class	
Class Epum	
C EDUR	
Gendin	
🗊 Interface	
🖄 Java Project	- 1
📽 Java Project from Existing Ant Buildfile	
曫 Java Working Set	
🕆 Package	- 1
😂 Source Folder	
🍪 Typesafe Enum	
🕨 🗁 Java Run/Debug	
V 🔁 JUnit	
🖹 JUnit Test Case	
🛍 JUnit Test Suite	
Java Emitter Templates	
🕨 🧁 Maven	
(?) < Back Next > Cancel Finish	

24. Select the location for the unit test in the src folder of our test fragment. And select the calculateForce method for testing:

🛞 New JUnit Test Case		
JUnit Test Case Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.		
O New JUnit <u>3</u> t	est 🖲 New JUnit <u>4</u> test	
Source fol <u>d</u> er:	com.acme.n2l.tests/src Browse	
Pac <u>k</u> age:	com.acme.n2l Bro <u>w</u> se	
Na <u>m</u> e:	NewtonsSecondLawTest	
<u>S</u> uperclass:	java.lang.Object Brows <u>e</u>	
Which method st	ubs would you like to	
	setUpBeforeClass() = tearDownAfterClass()	
	set <u>U</u> p() <u>t</u> earDown()	
	<u>c</u> onstructor	
Do you want to a	dd comments? (Configure templates and default value <u>here</u>)	
	<u>Generate comments</u>	
C <u>l</u> ass under test:	com.acme.n2l.NewtonsSecondLaw	
?	< Back Next > Cancel Finish	

Click "Finish"

25. Now we need to replace the template test code with our own. Add the following:

```
@Test
public void testCalculateForce() {
    MassAmount m = new MassAmount(1000, SI.KILOGRAM);
    AccelerationAmount a = new AccelerationAmount(2.5, SI.METRES_PER_SQUARE_SECOND);
    ForceAmount force = NewtonsSecondLaw.calculateForce(m, a);
```

```
assertEquals(2500, force.doubleValue(SI.NEWTON), 0.0001);
}
```

This method simply creates a 1000 kg mass, a 2.5 m/s2 acceleration and asks our N2L class to calculate the force. We then assert that the calculated force is 2500 Newtons (+/- 0.0001).

26. Save the file and Right click on it - then choose "Run As - > JUnit Plug-in Test". The test will run and you should see the JUnit results window appear:



27. You have now completed and tested your first simple UOMo project.

OK, so what....

Ok that's the basics out of the way now you can try a few things that demonstrate the power of UOMo and the Units of Measurement API...

- 1. Try changing the units used in your unit test for mass or acceleration to units of other quantities (i.e. seconds, metres, amperes, volts...). You will see the compiler error created because your unit is not the correct type... cool!
- 2. Same is true when we try to extract a value change the force.doubleValue line in the unit test to use units that are not a force... same, the compiler error because the unit is not for a force! cool!

This is important, it protects clients of your NewtonsSecondLaw API from making mistakes when they create values to pass to you. They can only pass in legitimate masses or accelerations.

As an example we can add another test method to our test case:

```
@Test
public void testWithOddUnits() {
    // We create a mass in US Pounds!
    MassAmount m = new MassAmount(100, USCustomary.POUND);
    // Now let's create a whacky acceleration unit of our own...
    @SuppressWarnings("unchecked") // we know this creates an acceleration!
    Unit<Acceleration> inch_per_square_second =
    (Unit<Acceleration>) USCustomary.INCH.divide(SI.SECOND).divide(SI.SECOND);
    // Note our N2L API does not know this unit at all!
    AccelerationAmount a = new AccelerationAmount(100, inch_per_square_second);
    ForceAmount force = NewtonsSecondLaw.calculateForce(m, a);
    // Yet our API is able to calculate the corresponding force without change
```

```
// Nice...
assertEquals(867961.6621451874, force.doubleValue(SI.NEWTON), 0.0000000001);
// Now let's try to get the result in a weird unit...
// The "pound-force" is a unit for Force in English engineering units
// and British gravitational units (http://en.wikipedia.org/wiki/Pound-force)
Unit<Force> poundForce = SI.NEWTON.multiply(4.448222);
// and we can now extract the result of our previous calc in that new unit!
assertEquals(3860886.16071079, force.doubleValue(poundForce), 0.000000001);
```

}

Nice - our API can handle weird and wonderful unit values as input and we can extract weird and wonderful unit values as output! Sweet!

