



# Motorola *WEAVR*

***UML 2.0 Weaver***  
*Add-In to Telelogic TAU G2*

## Programmer's Guide

*by*

*Thomas Cottenier*

thomas.cottenier@motorola.com

(847) 538 37 39

## 1. Installation Instructions

### Windows

Drop the AOM add-in in the 'addins' folder of the TAU installation directory

Add the TAU G2 installation directory in the system \$PATH environment variable

### Linux

Not yet supported

### Solaris

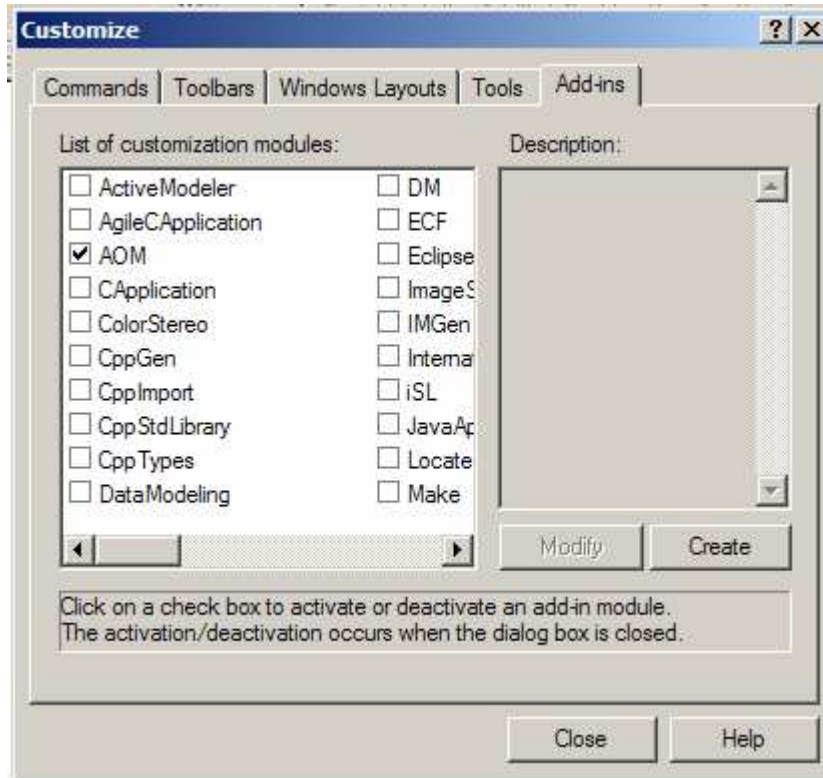
Two methods:

1. Drop the AOM add-in in the 'addins' folder of the TAU installation directory,
2. or run the `installWEAVR` script which will create a symbolic link in `$HOME/Telelogic/TAU\ 5.2/addins` that links to a pre-installed AOM add in.

**Note:** for the `comm.mot.com` domain the installation script is available in `/home/vdberg/installWEAVR`.

## 2. Activate the *WEAVR* AOM Add-in

1. Open the Add-ins tab, which is found in the **Tools** menu. Select Customize.
2. Check the AOM checkbox. Close the Add-ins tab.



**Figure 1.** Activating the WEAVR AOM Add-in

3. Save the workspace.
4. Close the workspace
5. Re-open the workspace.

**Note:** Operations 3, 4 and 5 are necessary for the following reason. The AOM Add-in needs to be active when you open your project, because it needs to capture the resources that are loaded in your project. Resources loaded into TAU before activating the Add-in will not be taken into account.

When re-opening the workspace, a new menu should appear with the following icons:



**Figure 2.** The WEAVR interface

The **Message Tab** in the Output window should also display messages similar to the listing of Listing 1.

```
WEAVR>> Getting Model Access...
```

```
Information: Session in ::[8w-1rLrIVn0LlGIayERkdtBL]: TMI0759: Loading file
C:\Program Files\Telelogic\TAU_2.5\uml2-itests\String01\String01.u2.
WEAVR>> Loading resource
Information: Session in ::[8w-1rLrIVn0LlGIayERkdtBL]: TMI0759: Loading file
C:\Program Files\Telelogic\TAU_2.5\examples\CommonRoutines.u2.
WEAVR>> Loading resource
Information: Session in ::[8w-1rLrIVn0LlGIayERkdtBL]: TMI0759: Loading file
C:\Program Files\Telelogic\TAU_2.5\examples\SuperDuperAspect1.u2.
WEAVR>> Loading resource
Information: Session in ::[8w-1rLrIVn0LlGIayERkdtBL]: TMI0759: Loading file
C:\Program Files\Telelogic\TAU_2.5\examples\SuperDuperAspectX.u2.
WEAVR>> Loading resource
WEAVR>> Initialize WEAVR Resource
Add-in module AOM activated.
Add-in module ModelVerifier activated.
Add-in module RTUtilities activated.
Add-in module SDL96Import activated.
```

**Listing 1.** Initialization of the WEAVR AOM Add-in

Messages outputted by the AOM add-in are preceded by the WEAVR>> tag.

The WEAVR messages displayed in Listing 1 indicate that the AOM add-in has successfully registered the resources loaded by TAU and has initialized successfully.

The AOMProfile library should also be loaded in the Model View, as displayed in Figure 3.

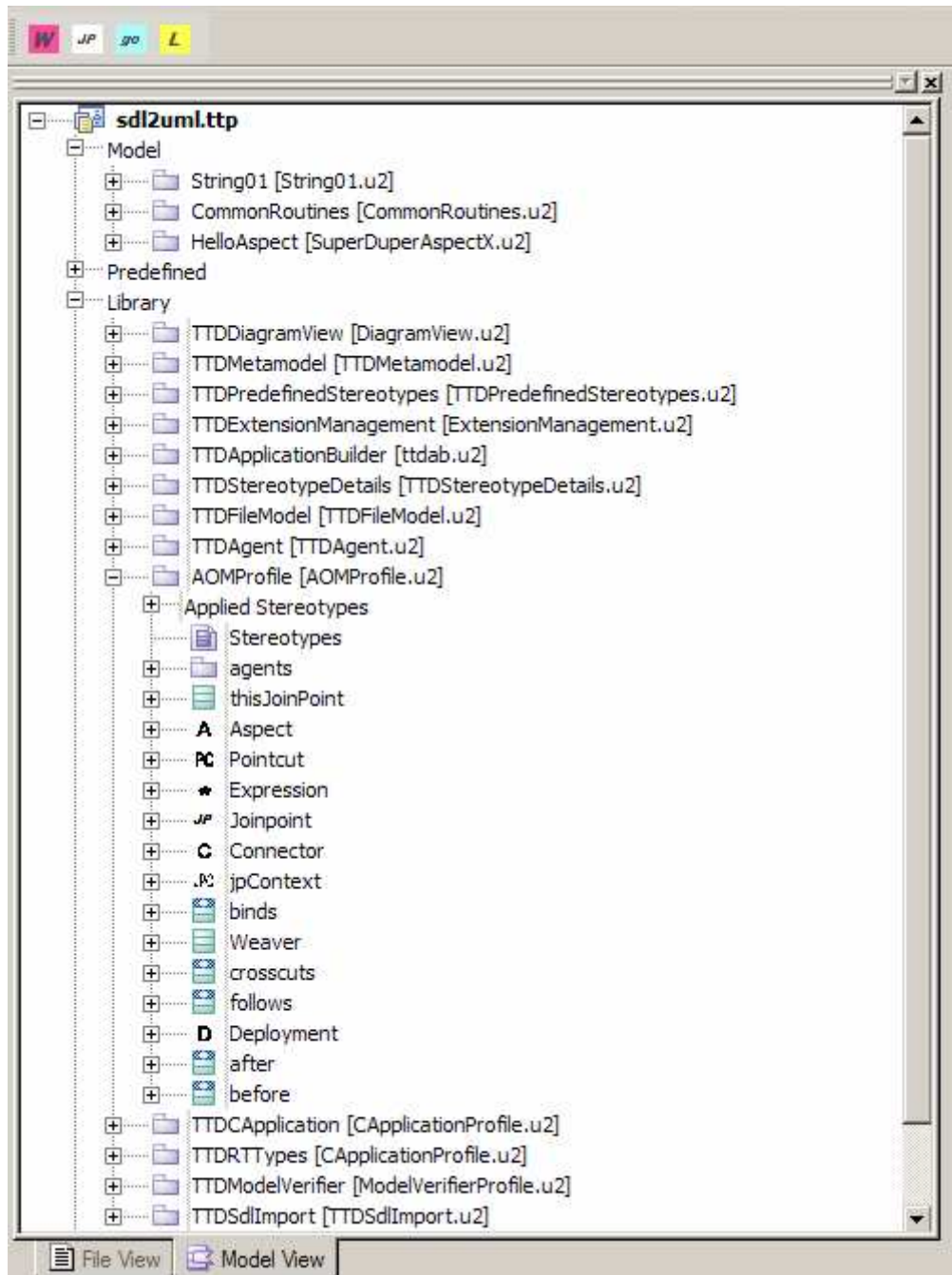





Figure 3. AOM Profile in the Model View.

### 3. WEAVR AOM Add-in Operations

The AOM add-in supports 3 distinct operations which are triggered by the buttons appearing in the WEAVR menu.

- Enter the WEAVR mode 
- Show Joinpoints 
- Weave Model 

### 3.1 Enter the *WEAVR* Mode

The Show Joinpoints button and the Weave Model button will only be activated after the WEAVR mode has been entered. The WEAVR mode ensures that weaving operations only proceed on a copy of the project. This restriction guarantees that the original model is never overwritten by a woven model.

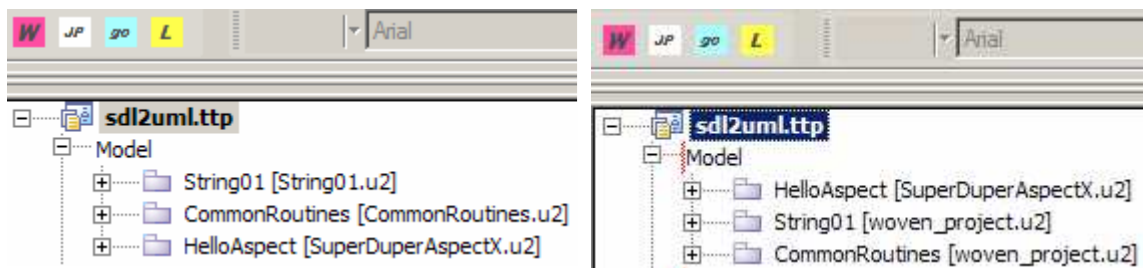
When entering the WEAVR mode, the AOM add-in takes a snapshot of the model in its current state, and makes a copy of the model, as well as all the entities on which it depends on. This copy is stored in a new resource, the **woven\_project.u2** file which is located in the same directory as the original model.

The corresponding project **woven\_project.ttp** and workspace **woven\_project.ttw** are also created in this directory.

```
WEAVR>> Entering WEAVR mode...
WEAVR>> loading String01...
WEAVR>> loading CommonRoutines...
WEAVR>> loading HelloAspect...
WEAVR>> Remove original entity String01
WEAVR>> Remove original entity CommonRoutines
WEAVR>> Entered WEAVR mode...
```

**Listing 2.** Entering the weaving mode

The WEAVR messages displayed in Listing 2 indicate that the AOM add-in has successfully copied the model resources and entered the WEAVR mode. Note that the structure of the project has been modified in the **Model View**, as displayed in Figure 4.



**Figure 4.** Model View before and after entering the WEAVR mode

Resources containing Aspects are not affected by this operation.

**Note:** Aspect weaving can affect both the model and the entities it depends on. To ensure that shared resources such as predefined libraries or model resources that are shared among development teams are not overwritten by Aspect weaving, the AOM addin imports all the dependencies of the model in the woven\_project.ttp project. The woven\_project.u2 resource file contains all the model entities (whether or not the original model contained multiple resource files) and all its dependencies.

### 3.2 Show Joinpoints

The Show Joinpoint button is only active after the project has entered the WEAVR mode.

The Show Joinpoints button triggers several successive operations:

1. Parse the Aspect Deployment Diagram
2. Parse the Aspects
3. Find the Joinpoints in the model
4. Annotate Joinpoints
5. Instantiate Connectors

These operations do not affect the semantics of the model. Operations 1, 2 and 3 gather the information required to perform the weaving. Operations 4 and 5 generate model elements that enable the developer to visualize the effects of the Aspects deployed in the model.

### 3.3 Weave Model

The Weave Model button is only active after the Show Joinpoints button has been triggered.

The weave model button drives the actual weaving of the model. Once the model has been woven, all the presentation elements of the model (diagrams) are destroyed. The woven model is automatically saved in the woven\_project.u2 file.

To edit the woven model, or perform **Code Generation** it is required to close the active project, and open the generated woven\_project.ttw workspace that has been created in the same directory as the project .ttw file.

In the woven\_project workspace, the presentation elements can be regenerated using the **Create Presentation** TAU utility.

**Generating a Diagram:** For example, in order to regenerate a State Chart Diagram, right-click on the state machine in the Model View. Select New in the Context Menu and select StateChart Diagram. Finally, drag and drop the State Machine *Implementation* from the Model View into the new State Chart Diagram.

## 4. WEAVR Add-in Utilities

The AOM add-in is shipped with 3 additional utilities that are useful for debugging purposes:

- XMLEncode
- Unparse
- Locate GUID

The XMLEncode and Unparse utilities are performed by the AOM.tcl Tcl script of the WEAVR distribution. The Tcl script is located in the **AOM/Script** directory and is directly customizable. The script is interpreted and can therefore be modified on the fly, without having to reload the project.

XMLEncode and Unparse are operations that are displayed in the Context Menu. These operations take as parameter the model entity on which the context menu is activated on (by right clicking on the model entity)

Locate GUID is triggered by the **L** button in the AOM Menu.

### 4.1 XMLEncode

XMLEncode creates a XML representation of the model entity on which it is called on. The XML representation is outputted in the Message Tab.

### 4.2 Unparse

Unparse creates a UML textual representation of the model entity on which it is called on. The UML textual representation is outputted in the Message Tab.

### 4.3 Locate GUID

The Locate GUID utility locates the model entity corresponding to a GUID. The Locate GUID button pops-up a window which prompts for a GUID identifier. The model entity corresponding to the GUID is located and highlighted in the model.

## 5. WEAVR Profile

The WEAVR profile defines extensions to UML 2.0 modeling elements for Aspect-Oriented Modeling.

The WEAVR Profile is located in the **AOM/Etc** directory. It can also be visualized in the Library directory of the Model View.

The WEAVR AOM Profile library has 3 different functions:

- Define the WEAVR Aspect-Oriented Modeling constructs
- Provide stereotypes to annotate the model elements that are affected by aspects (The Model Joinpoints).
- Provide a reflection library (thisJoinPoint)

### 5.1 WEAVR Aspect-Oriented Modeling constructs

The WEAVR uses two different types of Aspect-Oriented Modeling constructs:

1. Model Entities to define aspects.
2. Model Entities used to deploy aspects in a model and specify the interactions between Aspects.

#### 5.1.1 Aspect Model Entities

Figure 5 summarizes the Aspect Model Entities.

An **Aspect** (<sup>A</sup>) is a specialization of a Class. As such, it has all the properties of a regular class. It can define operations, attributes, ports, signals, timers, etc.

Aspects can also contain two special types of Operations: Pointcuts and Connectors.

A **Pointcut** (<sup>PC</sup>) is an operation that defines a query over model State Machine Entities. It specifies locations in the model state machines where behavior is injected. A pointcut is defined with respect to an **Expression**. An expression is a pattern over Operation, Signals or Timer signatures.

A **Connector** (<sup>C</sup>) is an operation that encapsulates the behavior that is injected at the locations in the model that match a pointcut expression.

The **Binds** dependency specifies a binding between Connectors and Pointcuts.

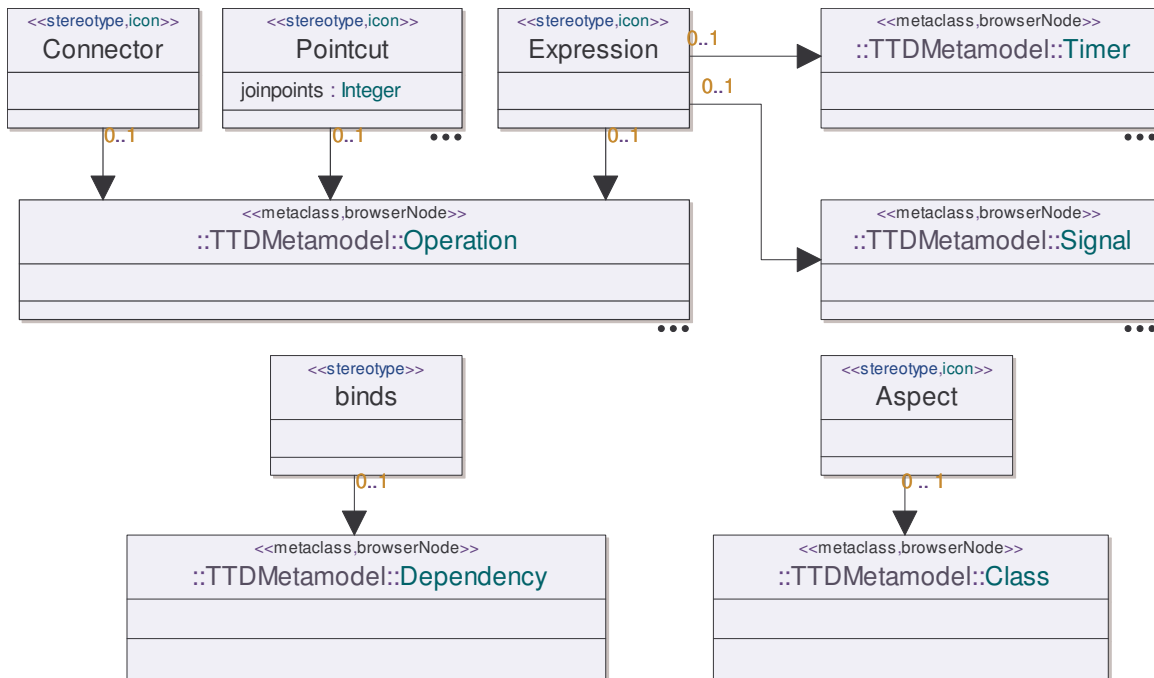


Figure 5. Aspect Model Entities

### 5.1.2 Aspect Deployment Entities

Aspect deployment entities are used to define which model Packages or Classes are subject to Aspect weaving.

A **Deployment** Diagram specifies the relationship between Aspects and Model Packages and Classes, as well as the relationships between Aspects.

The **Crosscuts** dependency specifies which Packages or Classes are subject to weaving for a specific Aspect.

The **Follows** dependency defines the order in which Aspects are woven in the model, when those interact with the model at the same locations.

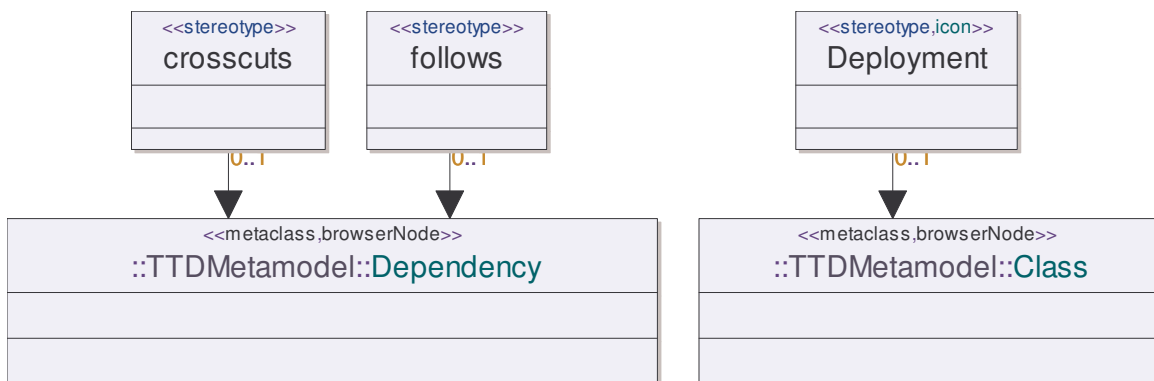


Figure 6. Aspect Deployment Entities

## 5.2 WEAVR Aspect Visualization Stereotypes

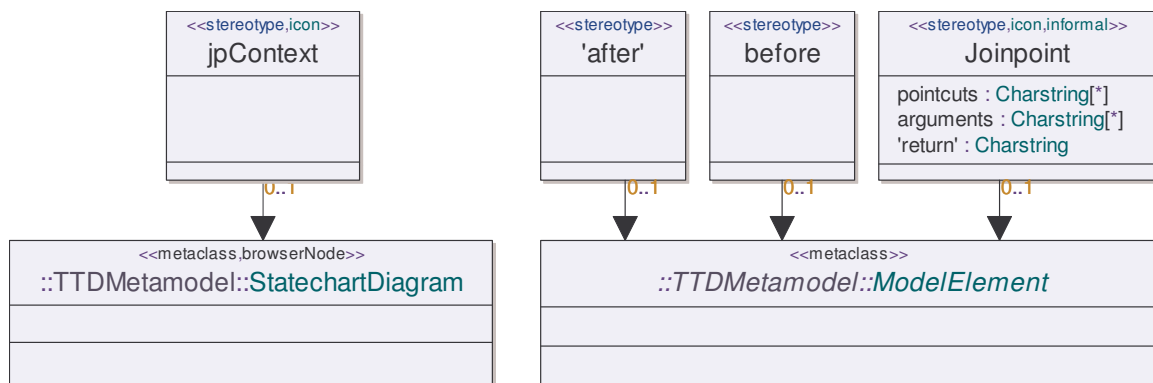
The Show Joinpoints ( **JP** ) *WEAVR* operation allows the user to visualize the locations in the model where Aspects apply.

Annotations are inserted in the model to indicate which Pointcuts and Aspects interact with the model at these locations.

**Joinpoints** ( **JP** ) are Model Elements that match a Pointcut Expression. More precisely, Joinpoints are either State Machine Actions or State Machine Transitions.

The **jpContext** ( **JPC** ) stereotype extends State Chart Diagrams that contains Joinpoints. State machines can include many State Chart Diagrams. It is therefore convenient to highlight those containing Joinpoints.

The **Before** and **After** stereotypes are used to delimit the entry point and the exits point of Transition Joinpoints.



**Figure 7.** Aspect Visualization Stereotypes

**Note:** The UML2.0 has not yet standardized the Action language. Actions are therefore not yet included in the TTDMetamodel. Yet, they are included in the TAU G2 Metamodel. For this reason, the stereotype extends ModelElement so that the stereotype can be applied to Actions.

## 5.3 WEAVR thisJoinPoint Reflective API

In most cases, the behavior that interacts with the model at the Joinpoints needs to be customized to the specific context in which it is bound to.

The AOM Profile library therefore includes a parameterized Class which provides utility methods to access this context. The different methods are documented in Section XX.

<div style="text-align: center;">thisJoinPoint &lt;class T &gt;</div>
<pre> + static getParameterTypeNames () : String + static getDeclaringTypeName () : String + static getName () : String + static getParameterNames () : String + static getThisClassName () : String + static inline ( 'in' : String) + static getObjectReference ( 'ref' : T) : String + static inlineObject ( a : String) + static getConnectorInstance () : T + static getThisClass () : TTDMetamodel::Class + static getTargetClass () : TTDMetamodel::Class + static getThis () : T + static getTarget () : T + static getTargetClassName () : String + static getSelf () : Pid + static getTarget () : Pid </pre>

Figure 8. thisJoinPoint reflective API

## 6. *WEAVR* Aspects

- 6.1 Aspects
- 6.2 Introductions (Inter-Type Declarations)
- 6.3 Pointcuts
- 6.4 Connectors
- 6.5 Binding Diagram
- 6.6 Aspect Deployment Diagram

## 7. Deployment of Aspects

## 8. Visualization of Aspect Effects

**9. Aspect Weaving**

**10. Code Generation from Aspect-Oriented Models**

**11.**