Collins CASE TA2 Isolator Transform

|  |
| --- |
| To illustrate the Isolator transform, we use a simple single-process example model, which can be found here:  https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Isolator/Simple%20Example  Two AADL packages are included:   * Untrusted\_Component.aadl – This is the initial model that includes a Resolute requirement for isolation of third-party components. * Isolated\_Component.aadl – This is the result of the Isolator transform on the initial model.   The Isolator transform can also be performed on the CASE Phase 1 UAV example model. Three versions of the model are available for reference:   * Initial model – This is the Phase 1 UAV model that includes an imported cyber requirement, which drives the isolation of the FlightPlanner component. The Initial model can be found here:   [https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Isolator/ UAV%20Example/Initial%20Model](https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Isolator/%20UAV%20Example/Initial%20Model)   * Transformed model – This is the Phase 1 UAV model after the Isolator transform has been applied. The Transformed model can be found here:   [https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Isolator/ UAV%20Example/Transformed%20Model](https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Isolator/%20UAV%20Example/Transformed%20Model)   * Test model – This is the Phase 1 UAV model containing several implementations for testing the correctness of the Resolute evaluation on the Filter transform. The Test model can be found here:   [https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Filter/ UAV%20Example/Test%20Model](https://github.com/loonwerks/CASE/tree/master/TA2/Model%20Transformations/Filter/%20UAV%20Example/Test%20Model) |

Processes, threads, and thread groups can be isolated inside a virtual machine. Note that to isolate software components using the Isolator model transformation, they must already be bound to a processor component. For example, the SW component is bound to the PROC component on line 60 of Untrusted\_Component.aadl (see Figure 1).

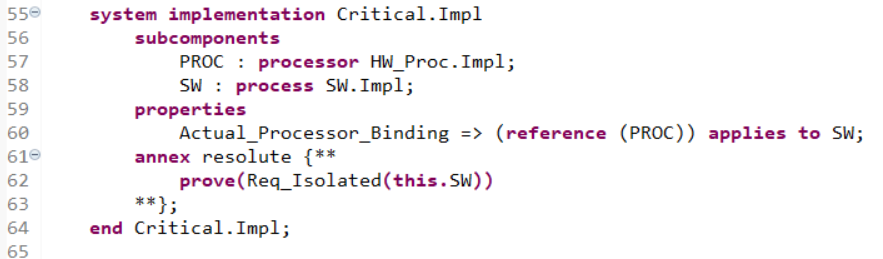


Figure 1. Line 60: Software process is bound to hardware processor component via Actual\_Processor\_Binding property association.

To apply the Isolator transform, select a process, thread, or thread group subcomponent in a component implementation (for example, in Untrusted\_Component.aadl, select the SW subcomponent on line 58). Note that currently the transformation can only be applied from within the OSATE text editor (future versions will enable applying the transformation from within the graphical editor). Click the CASE 🡪 Cyber Resiliency 🡪 Model Transformations 🡪 Add Isolator… menu item. A wizard will appear, as shown in Figure 2.

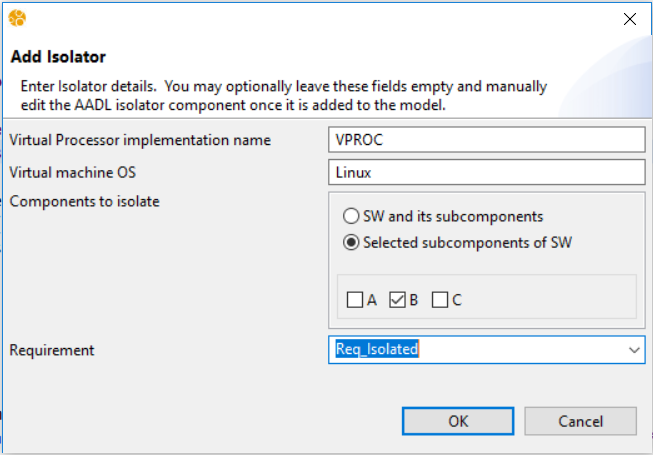


Figure 2. Add Isolator wizard.

The Isolator transform will create a new AADL virtual processor component and bind it to the same processor that the selected subcomponent was bound to. You can provide the name of the virtual processor subcomponent, or use the default name in the first field of the wizard. If the field is left blank, the default name will be used. Note that if the specified name already exists, a number will be appended to the name to make it unique within the containing component implementation.

The build process will package isolated components in a virtual machine. You can specify the virtual machine operating system, use the default name, or leave blank. The Isolator transform enables you to choose whether you would like to isolate the selected component and all of its subcomponents, or only specific subcomponents. Choosing to isolate only selected subcomponents will enable checkboxes for each subcomponent for selection/de-selection.

The requirement drop-down box lists all of the imported cyber-requirements from the TA1 tools. By specifying the cyber requirement that drives the isolator transformation, the appropriate assurance argument can be constructed for demonstrating the requirement was addressed correctly. A requirement does not need to be selected to insert the isolator, but it is highly recommended for construction of the proper system assurance case.

Clicking OK will close the wizard and apply the model transformation. A CASE\_Virtual\_Processor component type and component implementation is added to the AADL file, and a CASE\_Virtual\_Processor subcomponent is inserted into the component implementation containing the software component that was selected for isolation (see Figure 3).

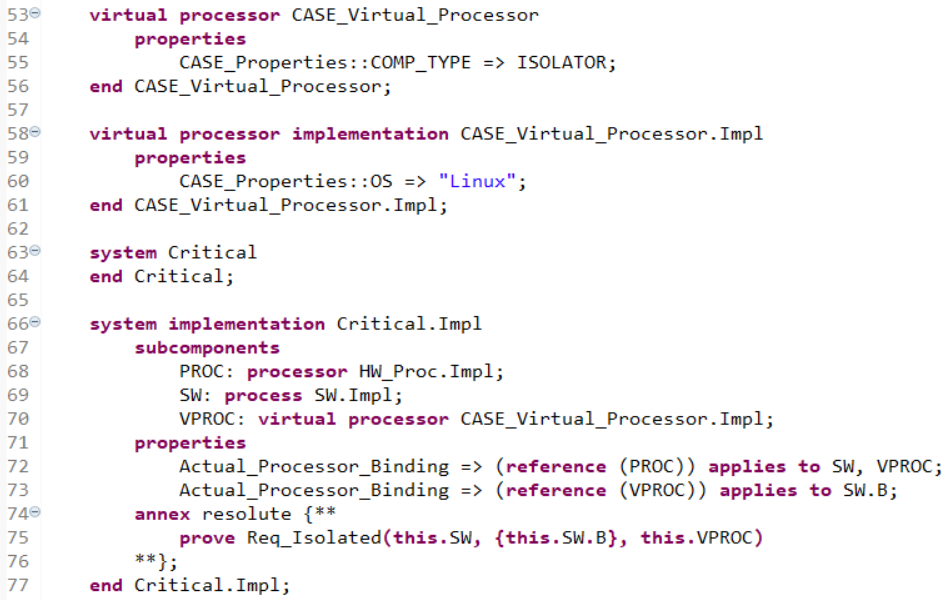


Figure 3. Line 53: CASE\_Virtual\_Processor component type; Line 58: CASE\_Virtual\_Processor component implementation; Line 70: CASE\_Virtual\_Processor subcomponent; Lines 72-73: updated processor bindings.

Virtualization is represented in AADL by binding a virtual processor component to a processor component, and then binding the isolated software components to the virtual processor. The transform will also remove existing bindings between the isolated software components and the processor component they were originally bound to. Note that per AADL semantics, if a component implementation is bound to a processor, that binding is also implicitly applied to that component’s subcomponents, unless a subcomponent has an explicit binding to a different processor.

### Design Assurance

As part of the transform, the requirement (specified in the model as a Resolute claim) will be updated with an add\_isolator subclaim from the CASE\_Model\_Transformations claim library (see Figure 4). This will provide assurance that the model transformation was performed correctly, and that the processor bindings are preserved throughout the remainder of system design, and through every step of the build process.

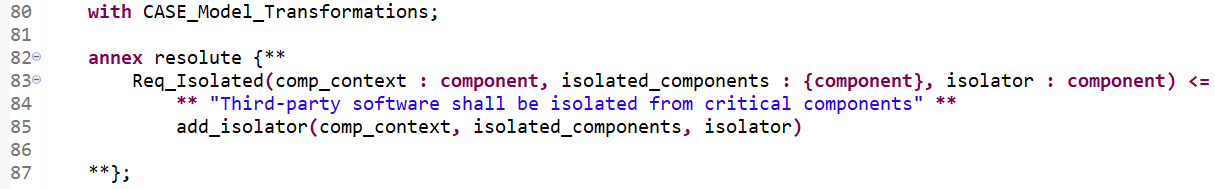


Figure 4. Isolator requirement in Resolute.