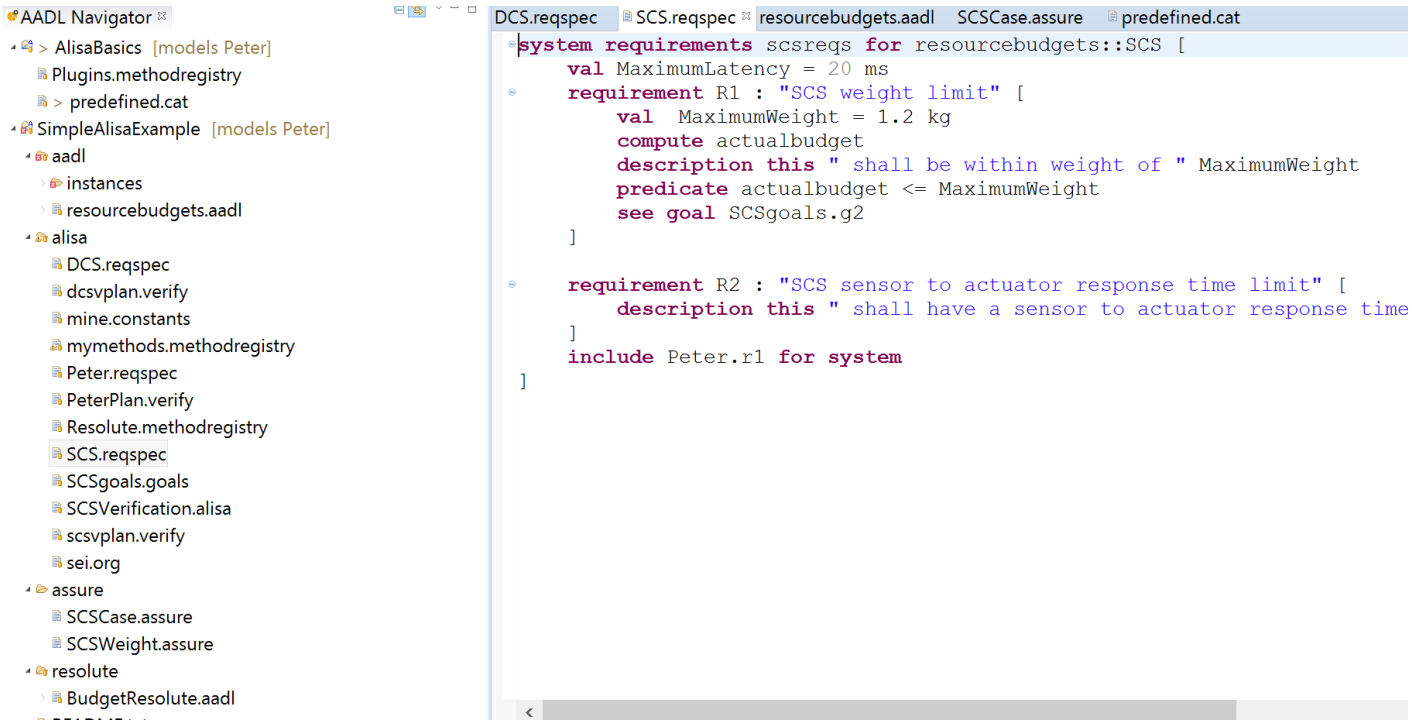
Getting Started

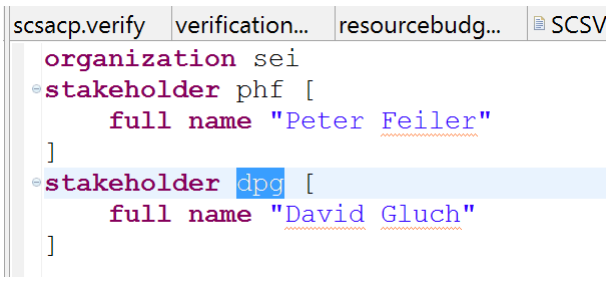
To get started, you should first download and install the Alisa feature with the plugins that support the different notations of Alisa as well as the assurance harness that executes the verification activities specified in verification plans on an instance of the AADL model for the system of interest. You then create a project as a Java project and then configure it with the AADL nature. You can create separate projects for your architecture models in AADL, for your requirement/verification specifications, and for other specifications such as Resolute libraries used in the verification - as shown in the example. Make sure you specify *Project References* through the *Preferences* menu. Select a project, select *Preferences* under the *Window* menu, (type *pref* into the search bar of the preferences dialog and then specify the projects in which this project will reference file content.



AADL, Alisa, and Resolute Files in Different Projects

You will create stakeholder goals and system requirements. You will do so in the context of an AADL model that represents the system of interest in its operational context by explicitly modeling all the entities in the operational environment that the system may be exposed to and interact with.

Stakeholders are defined as members of an organization. Each organization and its members is defined in a separate *.org* file.



Stakeholder Descriptions as Members of Organizations

Users can define a collection of stakeholder *goals* for a system . For each goal one or more stakeholders can be identified. In the process the user want to utilize a use case notation such as the URN Use Case Maps (UCM) - see jUCMNav.

```
stakeholder goals SCSgoals for resourcebudgets::SCS.tier0 [
    goal g1 : "Safety" [
        description "The system shall be safe."
        rationale "This is a control system, whose failure affects liv
        stakeholder sei.phf sei.dpg
    ]
    goal g2 : "Maximum weight" [
        description "The system shall stay within a speficied weight l
        stakeholder sei.phf
        rationale "The system is part of an aircraft."
    ]
]
```

Stakeholder Goals

System *requirements* are specified in a system *requirement specification*. They are traced to stakeholder goals and they identify the component type or implementation in the AADL model the requirement is associated with. Users can also specifies values as constants that can be used in descriptions and as operands in assert expressions. Typically, those are values that the user may want to change.

Requirements can be refined into requirements that are verifiable. All *leaf* requirements are expected to be verifiable.

DCS.reqspec | sei.org | scsvplan.verify | SCS.reqspec

```
system requirements scsreqs for resourcebudgets::SCS [
    val MaximumLatency = 20 ms
    requirement R1 : "SCS weight limit" [
        val  MaximumWeight = 1.2 kg
        compute actualbudget
        description this " shall be within weight of " MaximumWeight
        predicate actualbudget <= MaximumWeight
        see goal SCSgoals.g2
    ]

    requirement R2 : "SCS sensor to actuator response time limit" [
        description this " shall have a sensor to actuator response time wit
    ]
    include Peter.r1 for system
]
```

System Requirements in Context of an AADL Model

For each system the user specifies a verification plan. The verification plan specifies a claim for each requirement of the system of interest, identifying a collection of verification activities and making the argument of sufficient evidence if these verification activities applied to a system implementation pass. Users may impose an order to indicate that some are dependent on the successful completion of others, and specify that in case of an incomplete execution (*error, timeout*), or returning a negative result (*fail*) an alternative verification activity may be performed.

DCS.reqspec | sei.org | scsvplan.verify

```
verification plan scsvplan for scsreqs
[
    claim R1 [
        activities
            actualsystemweight : Plugins.MassAnalysis() [ phase Architectur
            Weightlimit: mymethods.assumeWithWeightLimit()
            MaxWeight : Resolute.verifySCSReq1(MaximumWeight)
    ]

claim R2 [
        activities
            responsetime : Plugins.FlowLatencyAnalysis()
            timing: Plugins.ResourceAllocationScheduling()

    ]
]
```

Verification Plan for System Requirements

Typically the user will work with an existing *registry* of *verification methods* for the OSATE plugins. Users can define additional registries for methods they wrote in Resolute, Java/Xtend.

Finally, the user will define the an *assurance plan* for a particular system. The assurance plan represents a configuration for verifying a system implementation that identifies the verification plans of those system components that the user is responsible for verifying. The user may define assurance tasks that specify selection criteria for a subset of the verification activities in the assurance plan that are to be performed as part of this assurance task. The selection criterion is expressed in terms of *selection category* tags that have been attached to verification methods or activities. From the assurance task specification an assurance task instance is created. This instance is used to drive the execution of the verification activities and to record the results as assurance *evidence*.

DCS.reqspec | sei.org | scsvplan.verify | SCS.reqspec | SCSCase.assure

```
case SCSCase
[
    tbdcount 0 successcount 3 failcount 6 errorcount 2
    model SCSCase.SCSPlan
    for resourcebudgets::SCS.tier0
        [
        tbdcount 0 successcount 3 failcount 6 errorcount
        claim scsreqs.R1
        [
            tbdcount 0 successcount 1 failcount 1 errorco
            verification scsvplan.actualsystemweight
            [
                executionstate completed
                resultstate success
                issues [
                    success "[A] Sum of weights 0.700 kg
                            "platform:/resource/SimpleAli
                ]
                tbdcount 0
            successcount 1
            ]
```

Outline

- System case SCSCase: (S3 F6 T0 E2 tbd0 EL0 TS0)
  - Model SCSCase.SCSPlan: (S3 F6 T0 E2 tbd0 EL0 TS0)
    - Claim R1: SCS shall be within weight of TBD (S1 F1 T0 E2 tbd0 EL0 TS0)
    - Claim R2: SCS shall have a sensor to actuator response time within org.osa
      - Evidence responsetime: Analysis of all end-to-end flows in a system insta
      - Evidence timing: Bind all threads to processors while maintaining schedu
    - Claim r1: test me (S0 F1 T0 E0 tbd0 EL0 TS0)
      - Evidence test: Test me[F] (S0 F1 T0 E0 tbd0 EL0 TS0)
    - Subsystem sensor1: (S0 F1 T0 E0 tbd0 EL0 TS0)
      - Claim r1: test me (S0 F1 T0 E0 tbd0 EL0 TS0)
    - Subsystem sensor2: (S0 F1 T0 E0 tbd0 EL0 TS0)
    - Subsystem actuator: (S0 F1 T0 E0 tbd0 EL0 TS0)
    - Subsystem dcs: (S0 F1 T0 E0 tbd0 EL0 TS0)

Generated Assure file and Outline View

Users then initiate the execution of all verification activities for an assurance case by opening the *assure* file and selecting the root element in the *Outline* view, bring up the context menu by right clicking, and invoking the *Verify All Evidence* command.

Note that multiple assurance tasks can be defined for the same system. Each task may focus on assuring different aspects of the system, as identified by the chosen selection categories. Similarly users can define assurance plans to include different combinations of verification plans for different subsets of system

components, leading to an incremental approach of verification.