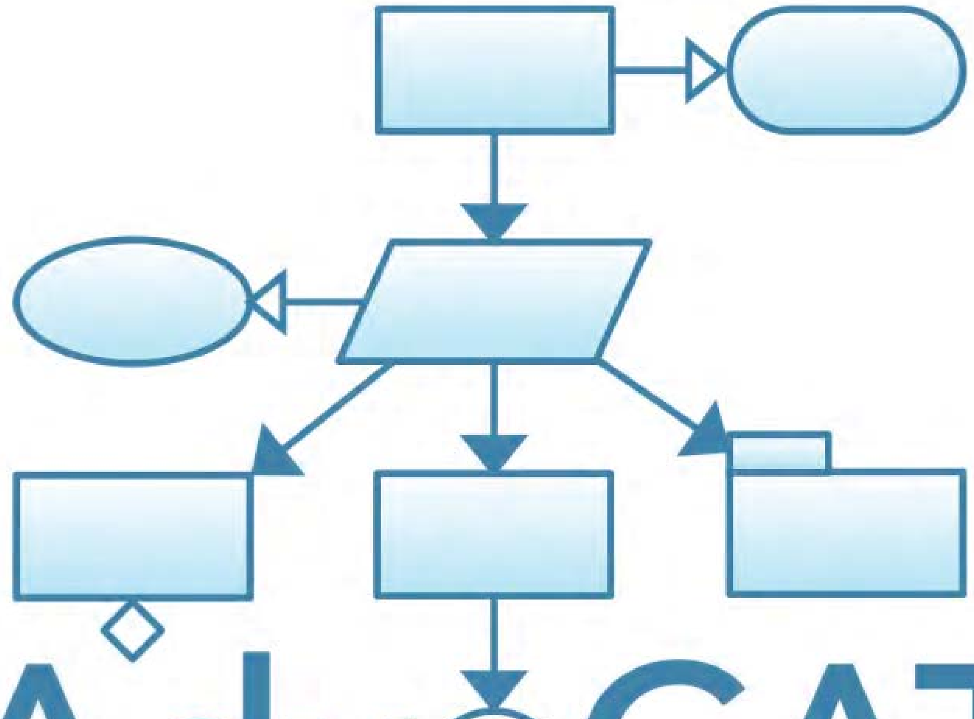


AdvoCATE

Assurance Case Automation Toolset

User Guide

Contents

1. Introduction

Introduction

- AdvoCATE Basic Vision
  - To automate the creation, manipulation and management of large-scale assurance cases based on a formal theory of argument structures

- Main purpose
  - Creating and manipulating argument structures for safety/assurance cases using the Goal Structuring Notation (GSN)
  - A test bed and proof-of-concept for the formal theory of argument structures

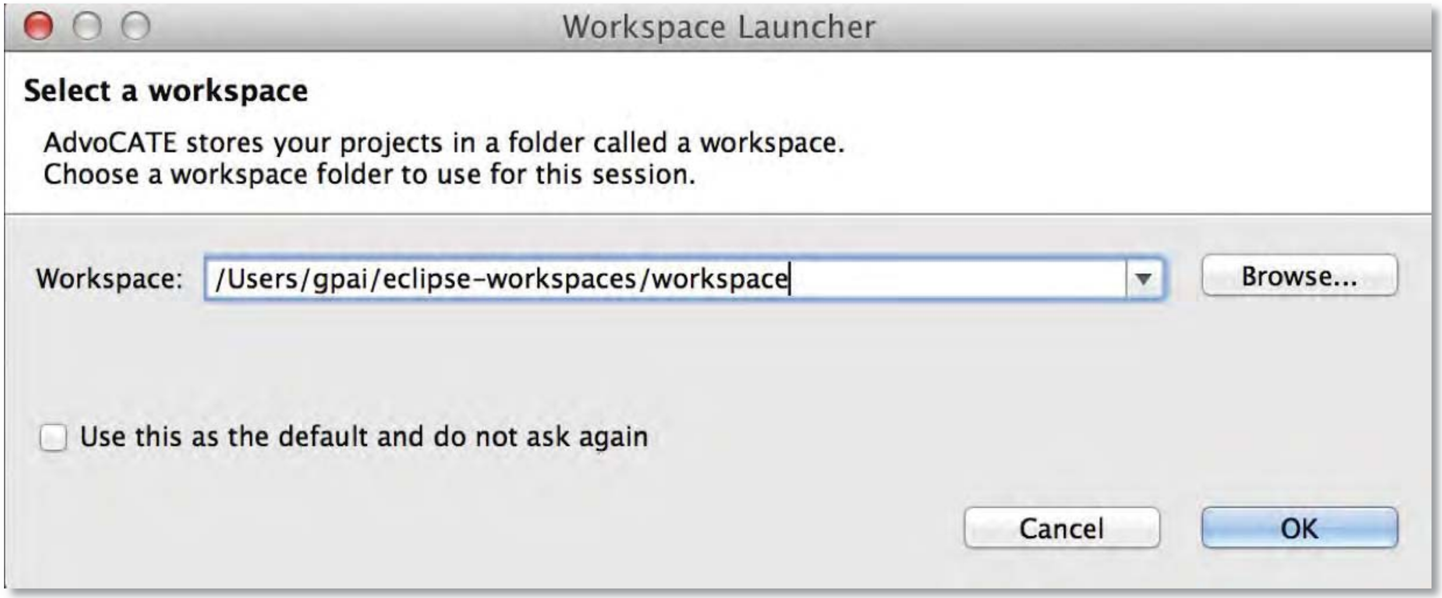
- AdvoCATE is available for
  - Windows 7
  - Macintosh OS X
  - Linux

- Eventually, AdvoCATE will
  - Serve as a dashboard for safety related information
  - Provide an infrastructure for safety decisions and management

2. Installation, Launch & Configuration

Installation

- Unzip the installation archive into your preferred folder
- Launch the executable file
- On launch, specify a path in the workspace launcher dialog box to create a new workspace



Workspace Launcher

Select a workspace

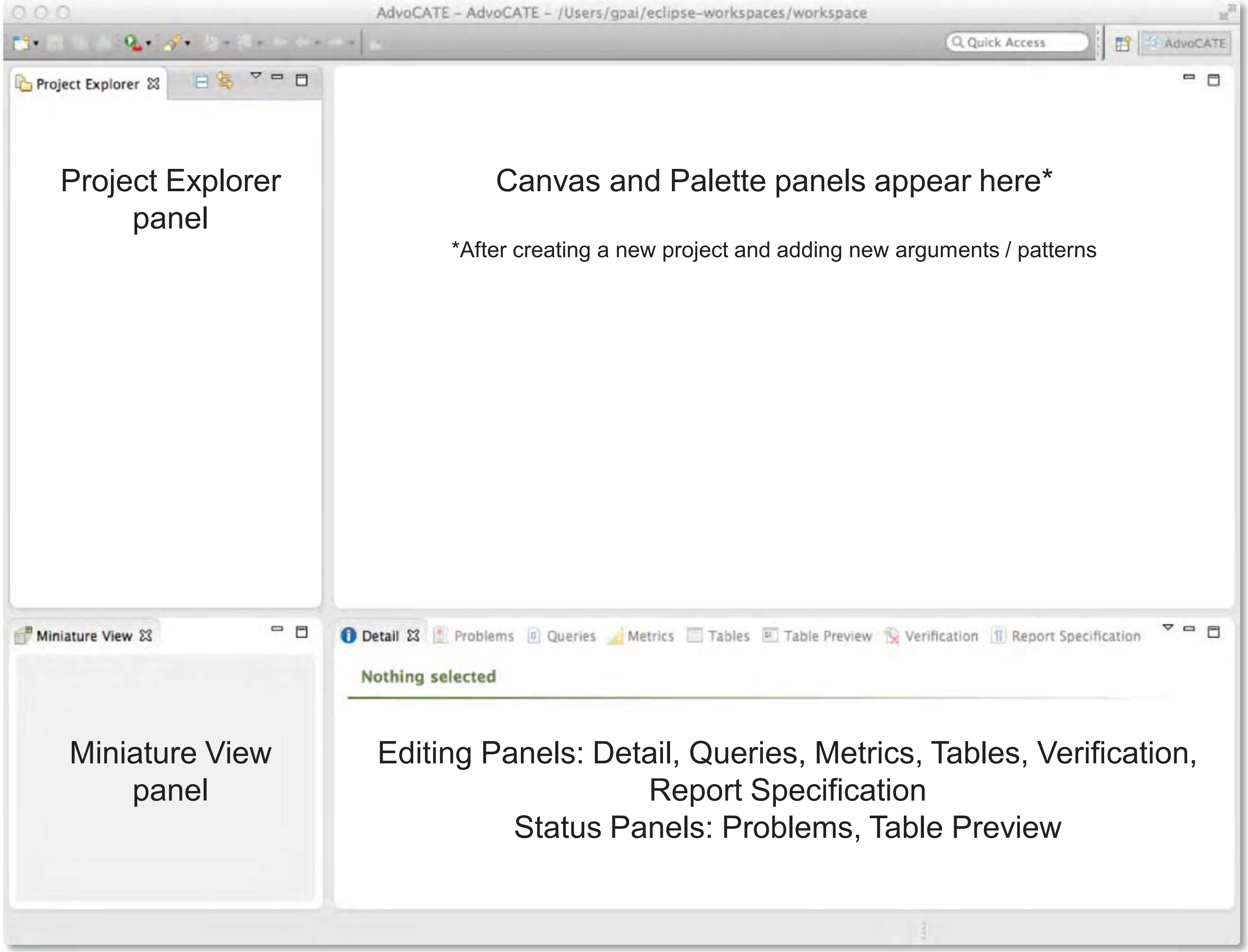
AdvoCATE stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /Users/gpai/eclipse-workspaces/workspace

□ Use this as the default and do not ask again

- A workspace is a folder that AdvoCATE uses to cache tool settings and the state
  - Workspaces can be periodically deleted to remove cached settings
- AdvoCATE will launch in the AdvoCATE *perspective*
  - The AdvoCATE perspective is a customized arrangement of IDE windows which you will use to create, manipulate and manage argument structures and patterns

AdvoCATE Perspective



Nothing selected

Project Explorer panel

Canvas and Palette panels appear here*

*After creating a new project and adding new arguments / patterns

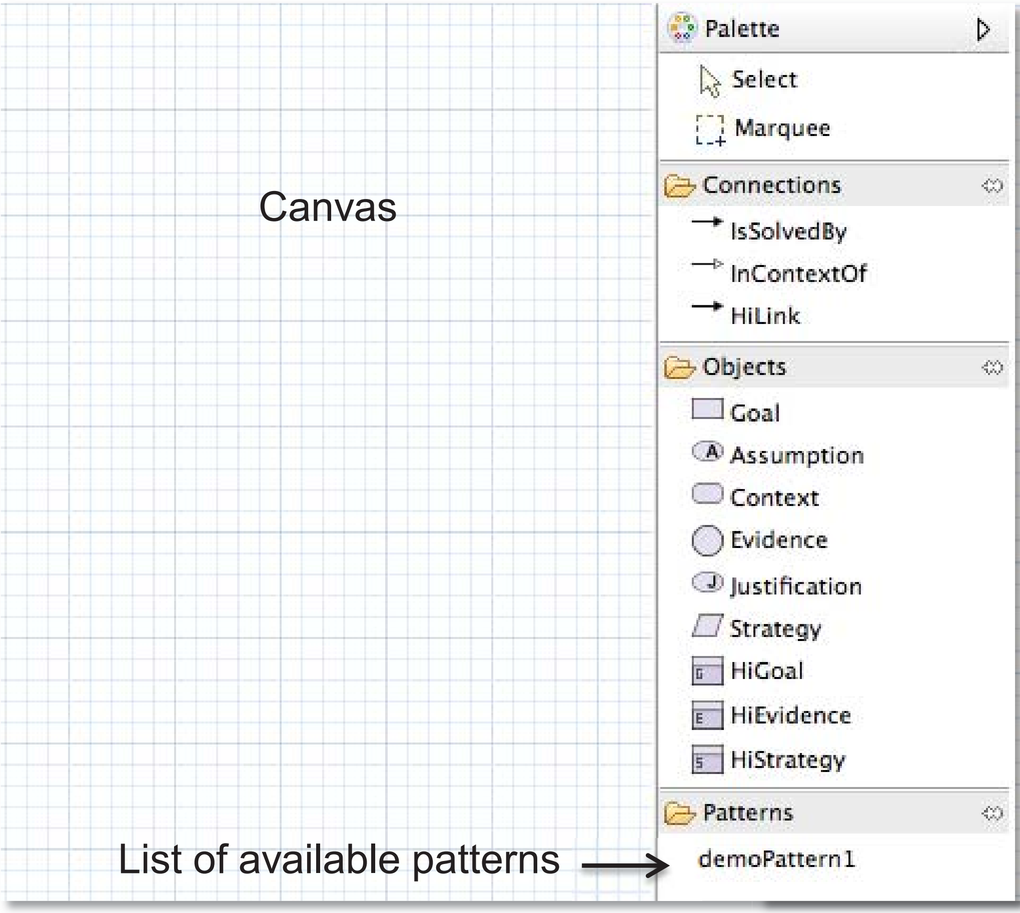
Miniature View panel

Editing Panels: Detail, Queries, Metrics, Tables, Verification, Report Specification

Status Panels: Problems, Table Preview

AdvoCATE Perspective – Canvas and Palette

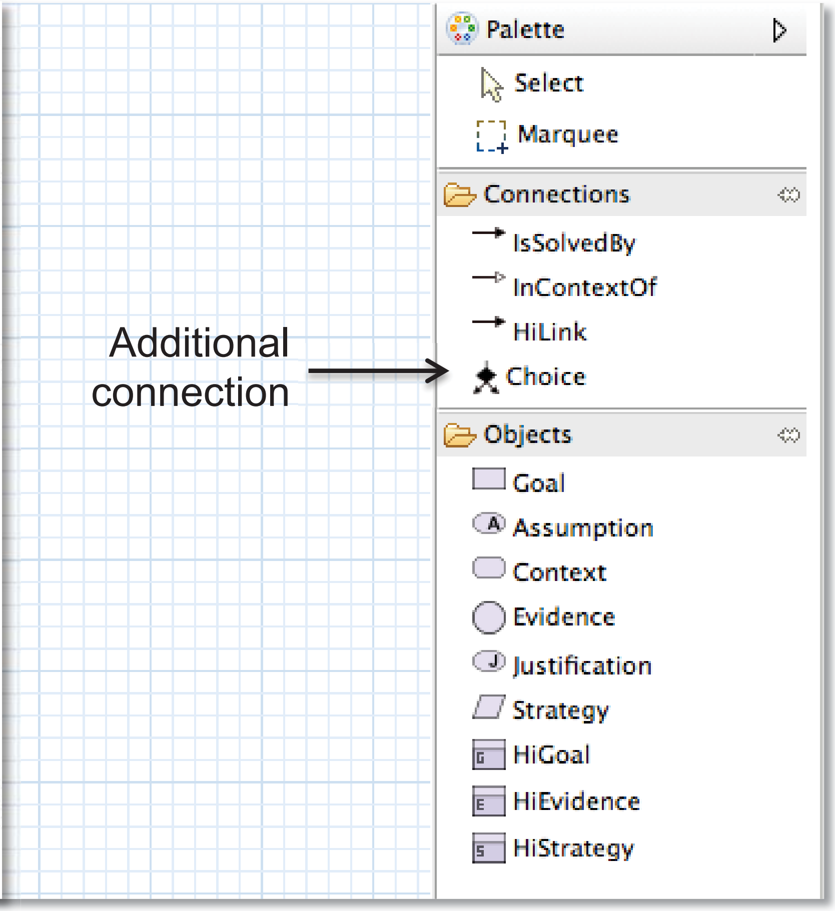
Canvas and Palette for Arguments



Canvas

List of available patterns

Canvas and Palette for Patterns

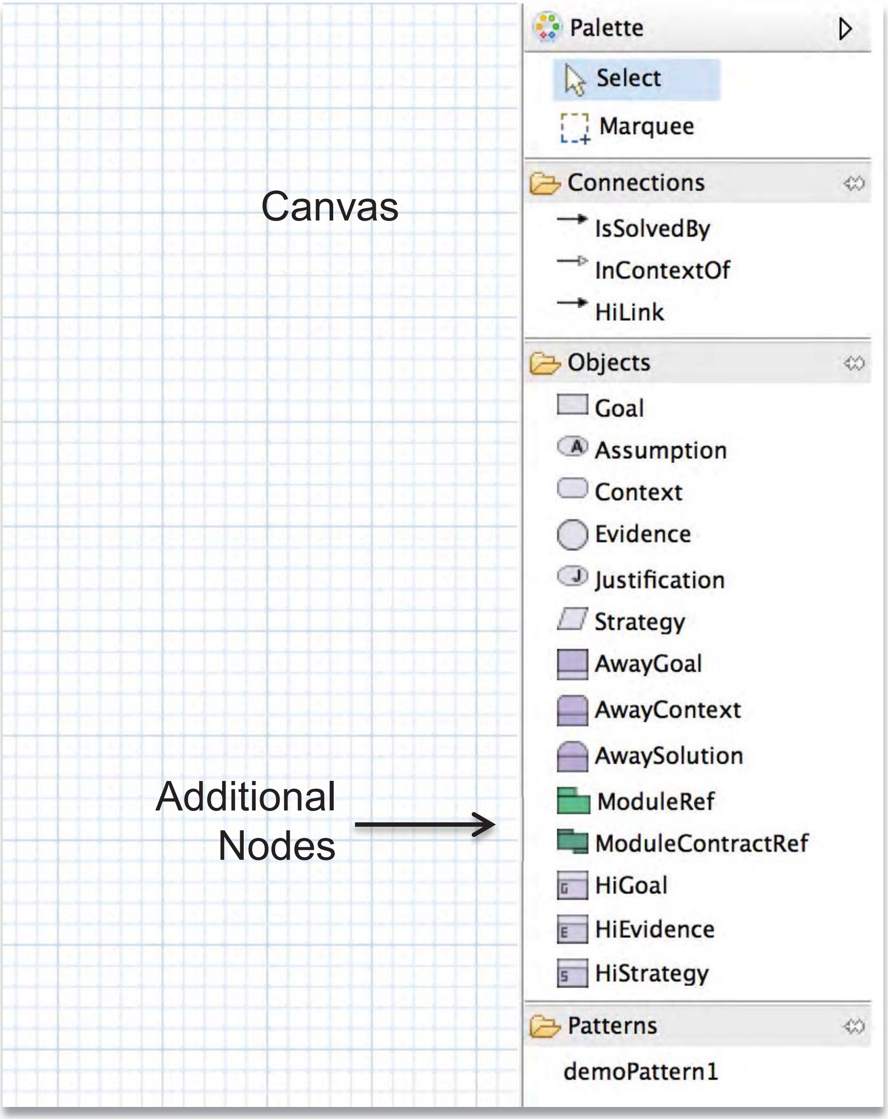


Additional connection

- Drag and drop objects and connections from the palette onto the canvas
  - The palette for arguments lists available patterns, i.e., a pattern library
  - The palette for patterns contains one additional *choice* connection

AdvoCATE Perspective – Canvas and Palette

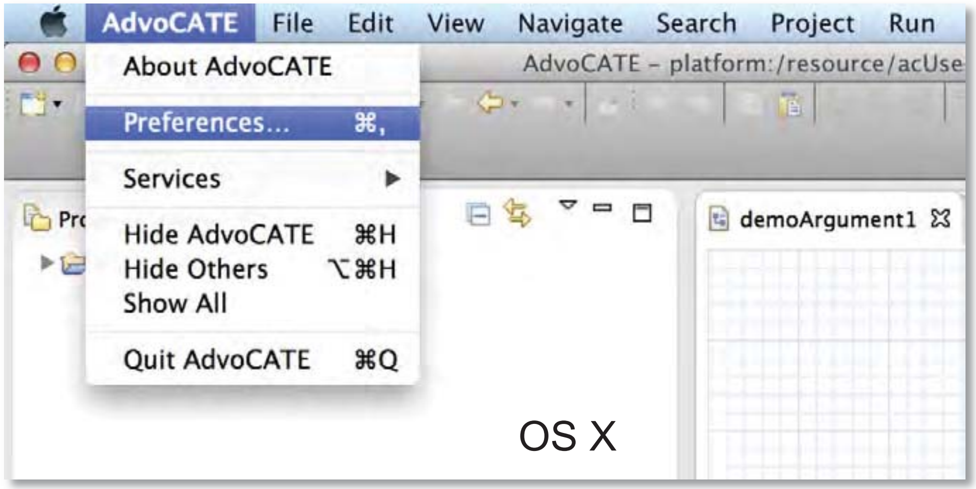
Canvas and Palette for Modules and Module Contracts



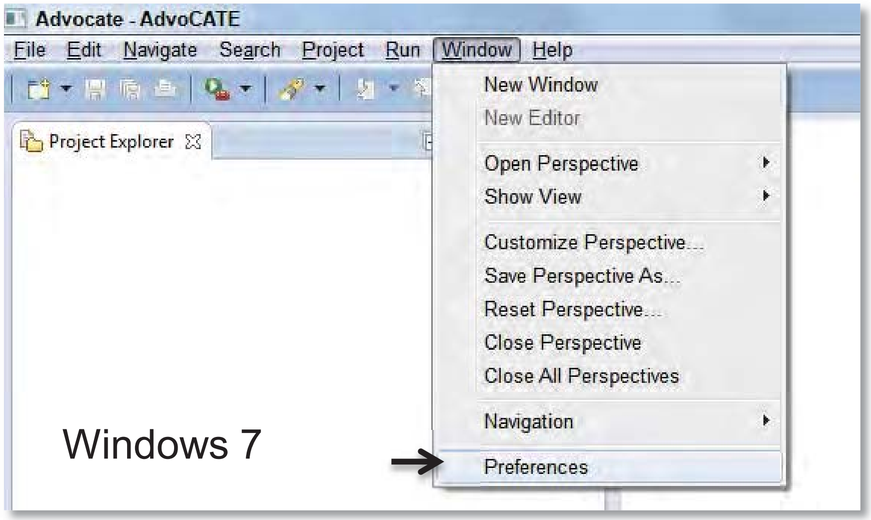
- The palette contains additional node elements for module and contract module diagrams
  - Module references
  - Contract module references

- As before, drag and drop objects and connections from the palette onto the canvas

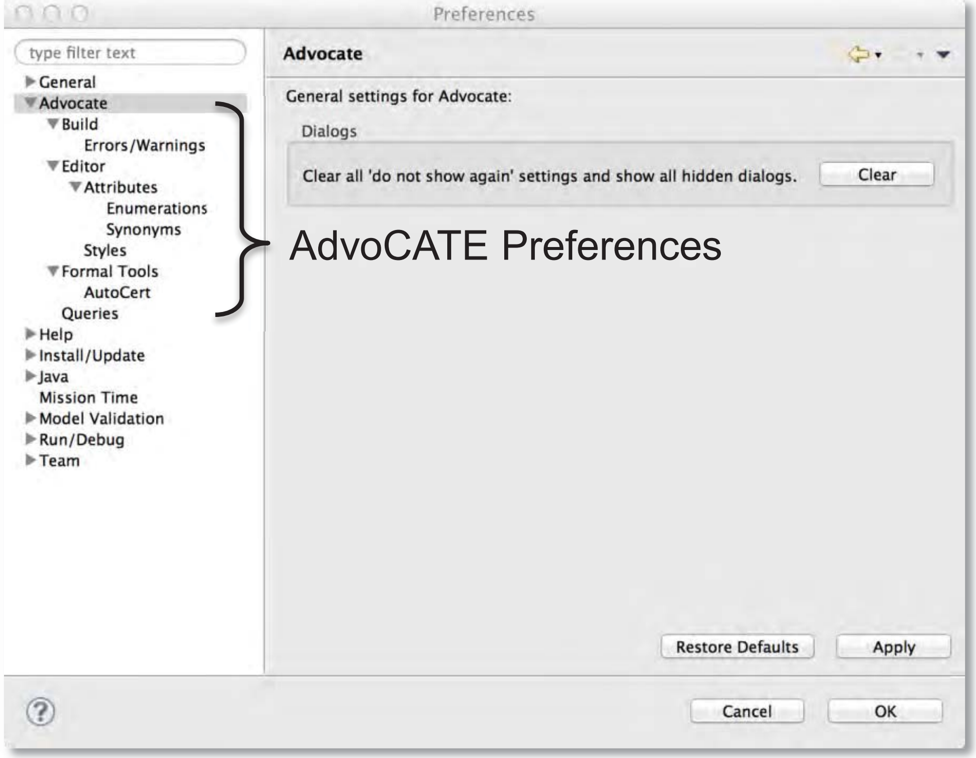
AdvoCATE Preferences



OS X



Windows 7



General settings for Advocate:

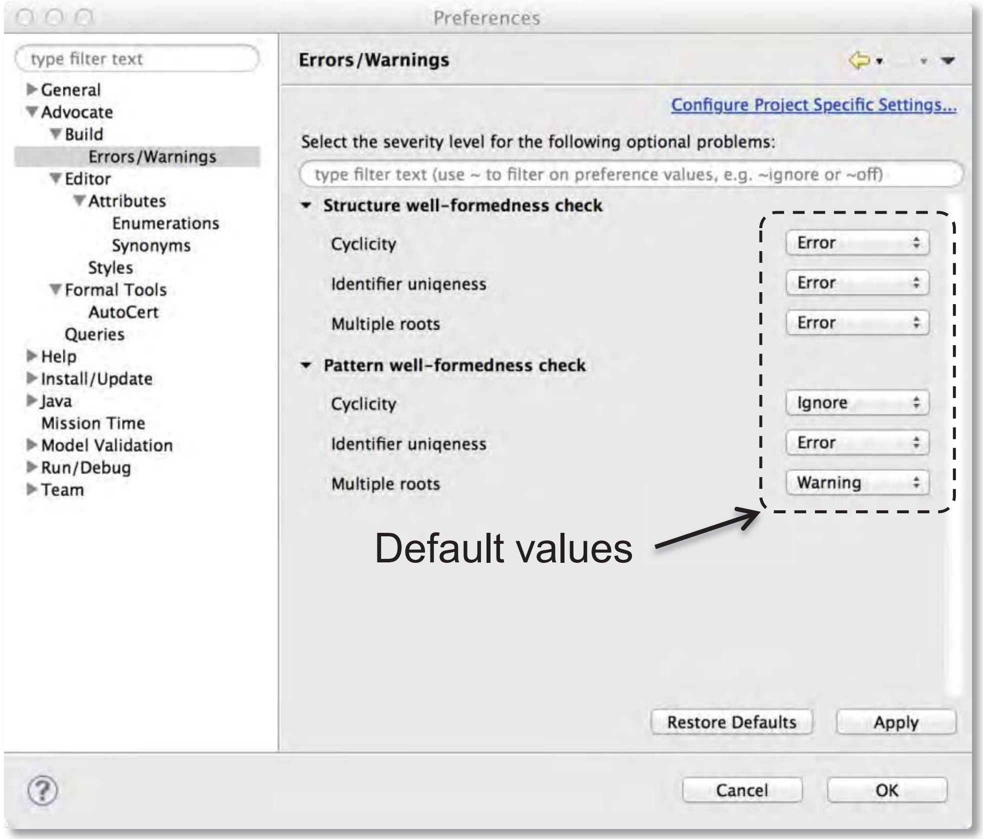
Clear all 'do not show again' settings and show all hidden dialogs.

AdvoCATE Preferences

- AdvoCATE preferences allow the user to configure tool settings with respect to
  - Errors / Warnings
  - Styles of the graphical elements
  - Metadata (described later)
  - Queries*
  - Formal tool integration*
- Other settings elements shown in the *Preferences* window are not currently relevant for AdvoCATE

*Not covered in this user guide

Preferences – Global Errors / Warnings



Select the severity level for the following optional problems:

Structure well-formedness check

- Cyclicity: Error
- Identifier uniqeness: Error
- Multiple roots: Error

Pattern well-formedness check

- Cyclicity: Ignore
- Identifier uniqeness: Error
- Multiple roots: Warning

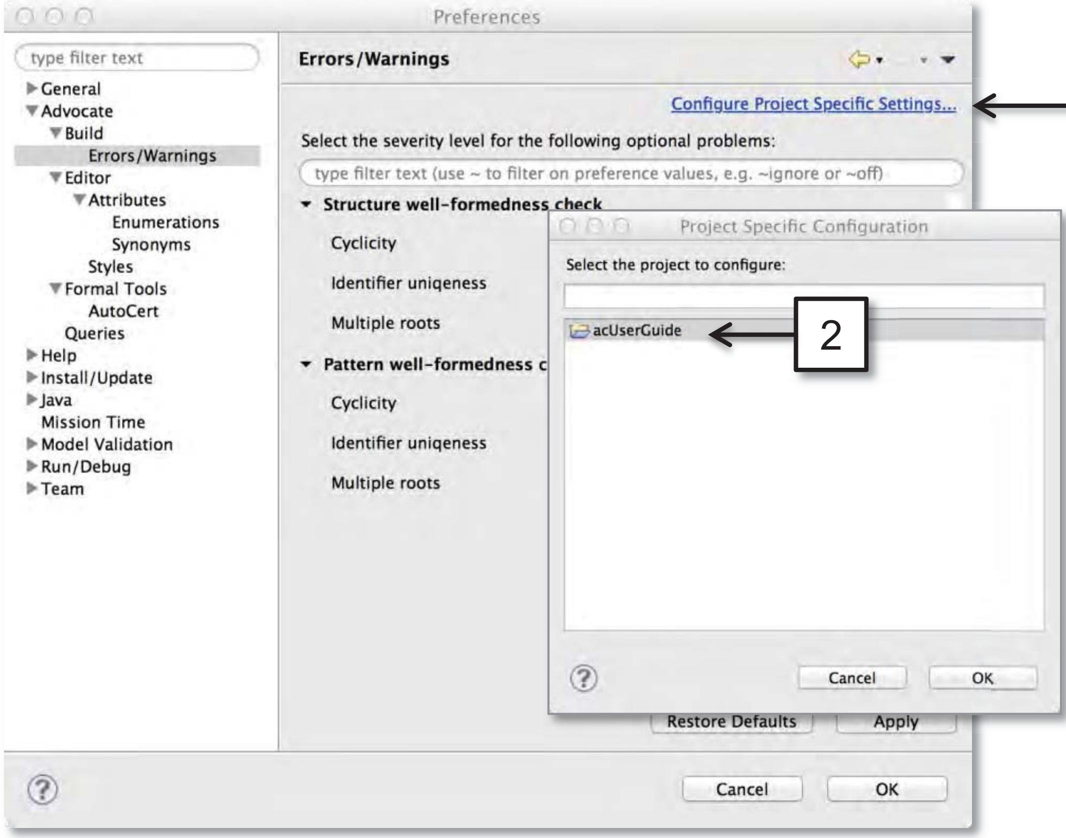
Default values

Note: These preferences apply globally to all projects in the workspace

- AdvoCATE checks well-formedness upon saving an argument structure
- Currently, 3 well-formedness checks are available for arguments and patterns:
  - Cyclicity: Presence/absence of cycles in the structure*
  - Identifier uniqueness: Unique node identifiers
  - Multiple roots: Presence of only a single root node
- Each check has 3 possible values:
  - Error
  - Ignore
  - Warning
- To alter the defaults, select a value from the drop-down list
- Outcomes of the checks are displayed in the Problems panel

*Cyclic structures are permitted in patterns

Preferences – Project Specific Errors / Warnings



Select the severity level for the following optional problems:

type filter text (use ~ to filter on preference values, e.g. ~ignore or ~off)

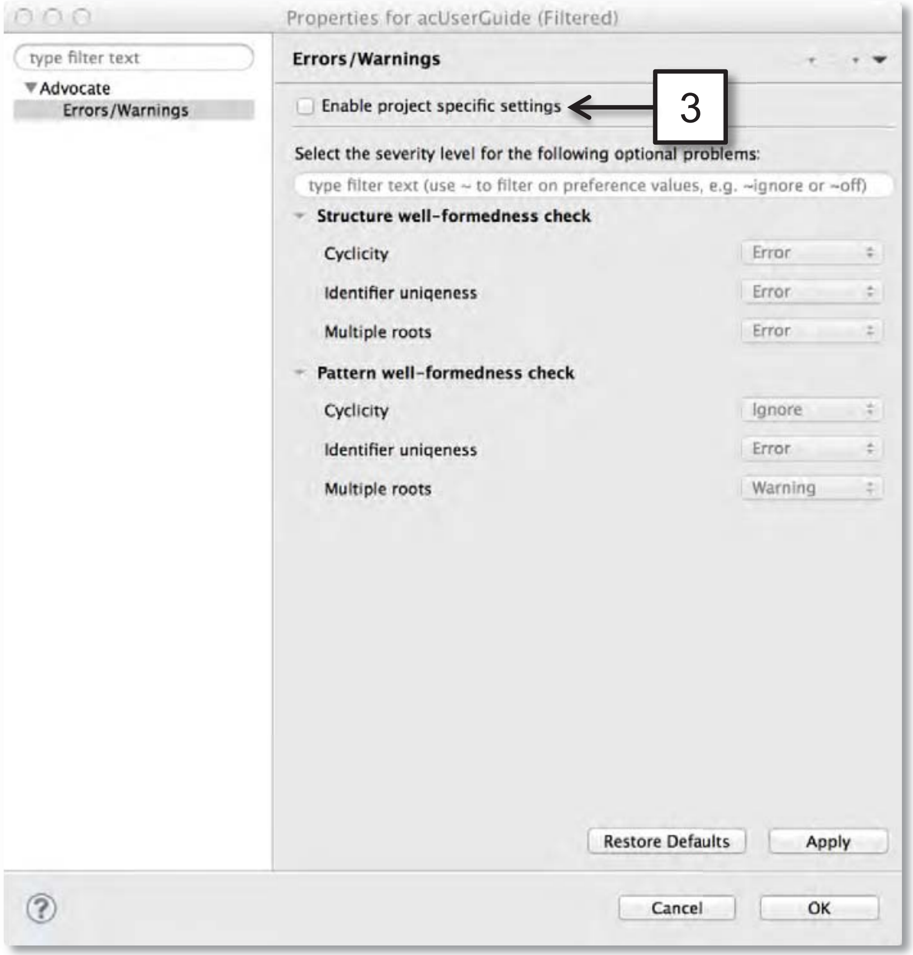
Structure well-formedness check

- Cyclicity
- Identifier uniqeness
- Multiple roots

Pattern well-formedness c

- Cyclicity
- Identifier uniqeness
- Multiple roots

Select the project to configure:



□ Enable project specific settings

Select the severity level for the following optional problems:

type filter text (use ~ to filter on preference values, e.g. ~ignore or ~off)

Structure well-formedness check

Cyclicity	Error
Identifier uniqeness	Error
Multiple roots	Error

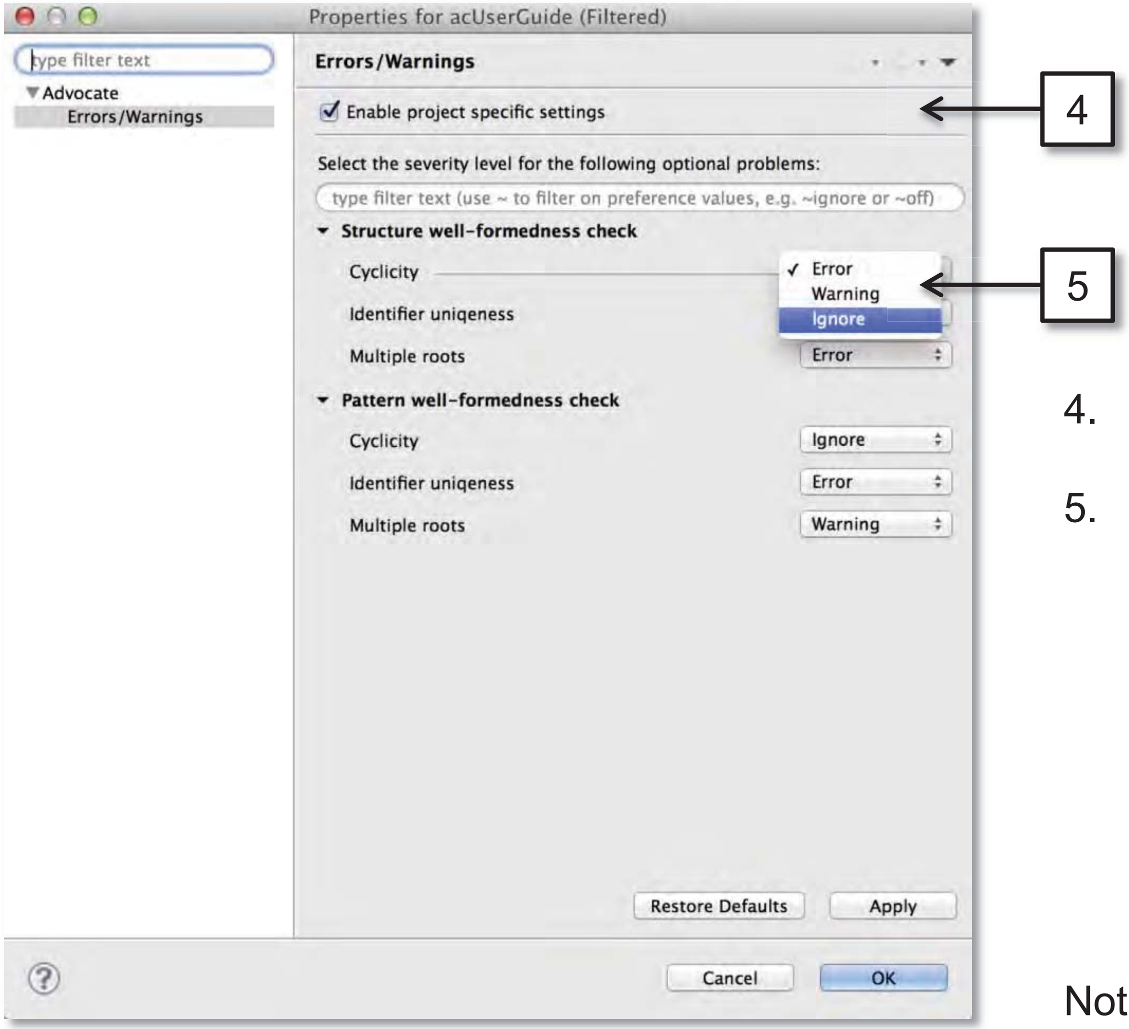
Pattern well-formedness check

Cyclicity	Ignore
Identifier uniqeness	Error
Multiple roots	Warning

To override global preferences and configure project-specific settings

1. Select *Configure Project Specific Settings*
2. Select a project from the listed projects
3. By default, project specific settings are disabled

Preferences – Project Specific Errors / Warnings (Cont'd.)



Properties for acUserGuide (Filtered)

type filter text

- Advocate
  - Errors/Warnings

Errors/Warnings

☑ Enable project specific settings

Select the severity level for the following optional problems:

type filter text (use ~ to filter on preference values, e.g. ~ignore or ~off)

Structure well-formedness check

Cyclicity	✓ Error Warning Ignore
Identifier uniqeness	
Multiple roots	Error

Pattern well-formedness check

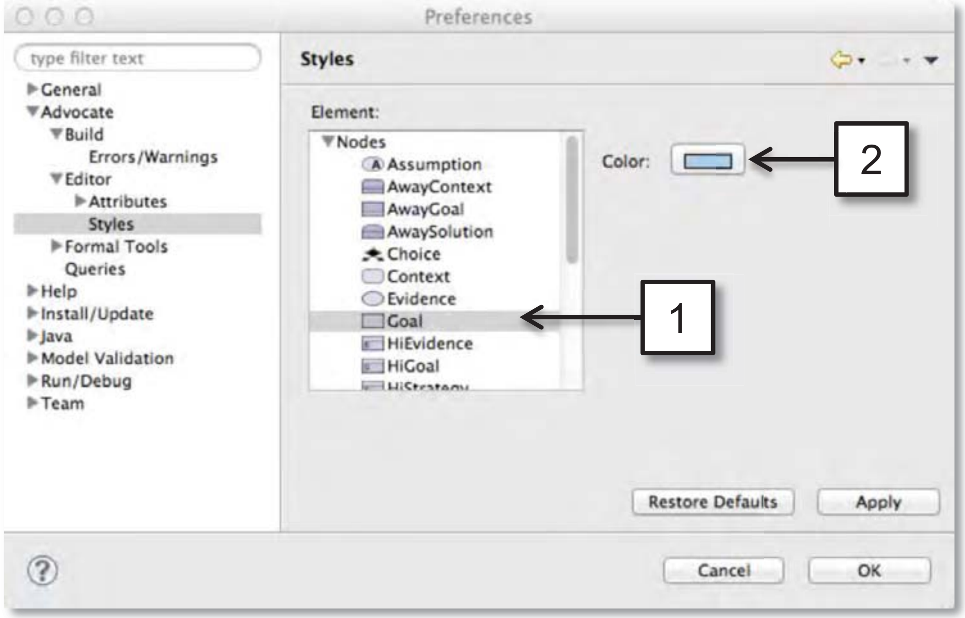
Cyclicity	Ignore
Identifier uniqeness	Error
Multiple roots	Warning

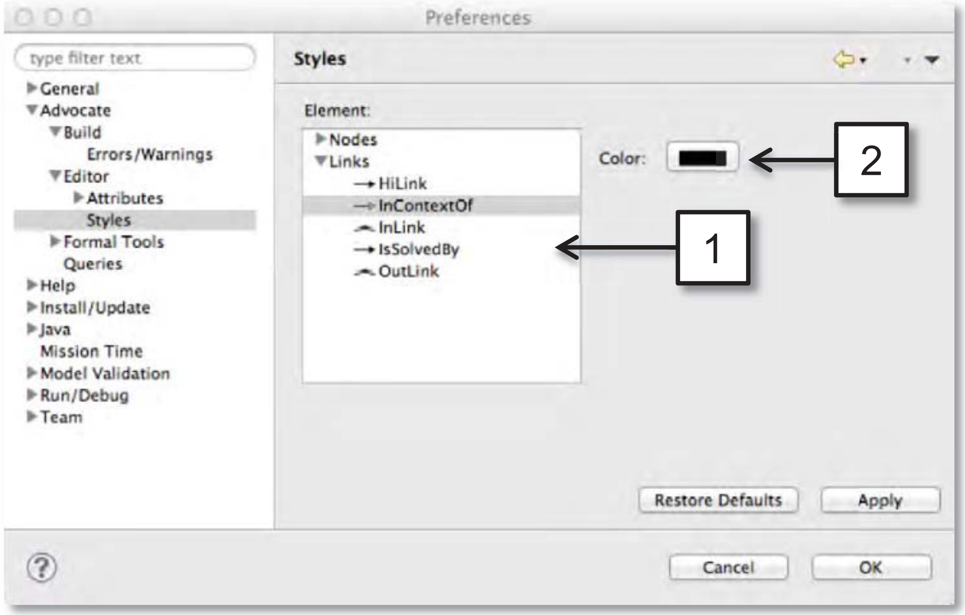
4. Enable project specific settings by selecting the check box
5. Update the default values by selecting a value from the drop-down list. As in the global preferences, project-specific well-formedness checks have 3 possible values
   - Error
   - Ignore
   - Warning

Note:

These preferences can also be accessed through Project properties

Preferences – Styles





- Nodes are colored blue by default
- Links are colored black by default

- To change the default value
  1. Select a node / link in the *Element* list
  2. Then click the *Color* button to choose a color

3. Projects

Overview

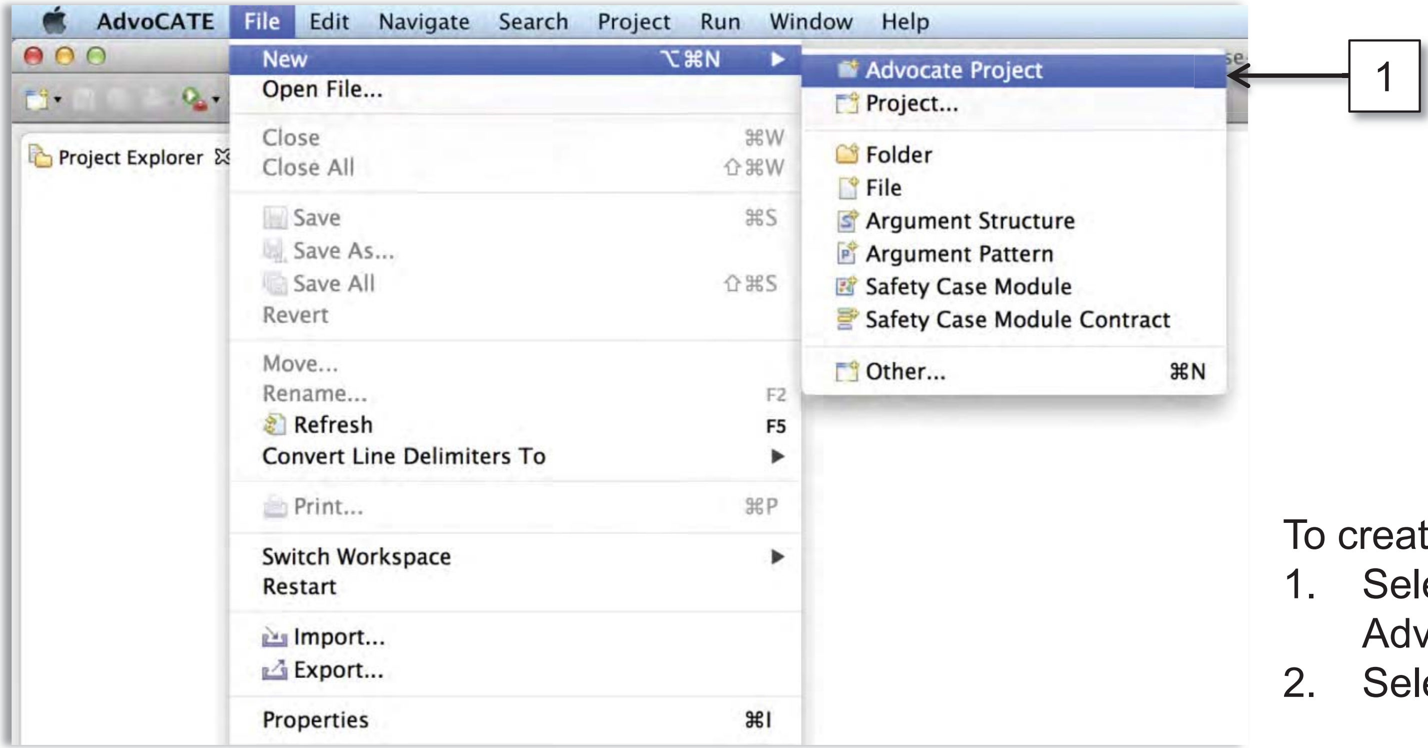
- Create an AdvoCATE *Project* to organize a safety / assurance case

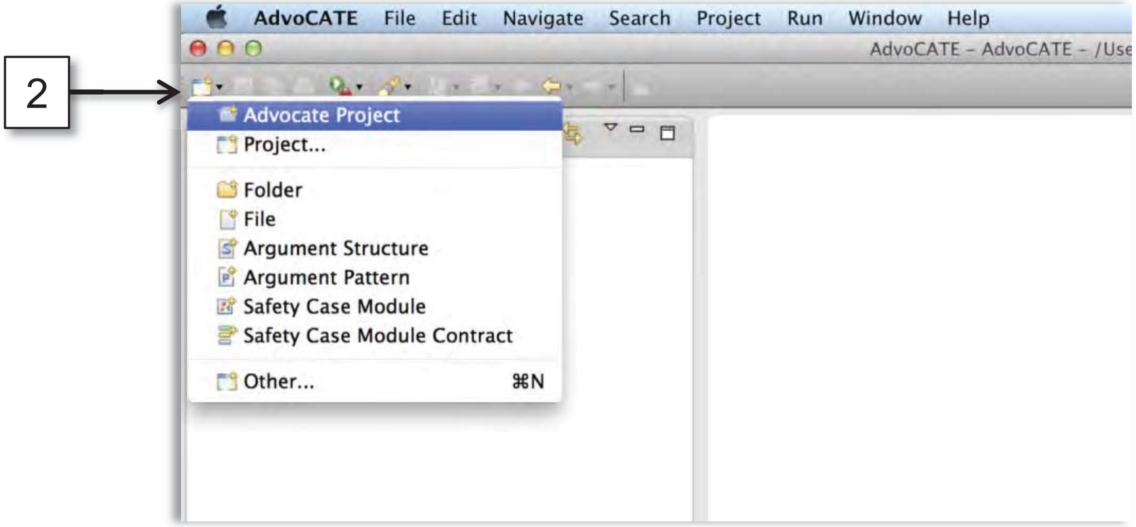
- Projects can contain
  - Diagram file types with the extension `.argument`
    - These diagrams can be
      - (Non modular) Argument structures
      - Module diagrams: An argument structure within a module
      - Contract module diagrams: An argument structure within a contract module

  - Diagram file types with the extension `.pattern`
    - These diagrams are non modular argument patterns

  - Auto-generated output, e.g., Views, Reports, Metrics
  - Folders
  - Other file types referenced by nodes in the argument structure representing
    - Documents: e.g., PDF files, MS Word Documents, Text files, etc.
    - Parameter values for pattern instantiation, e.g., in CSV files
    - Spreadsheets, e.g., MS Excel files
    - Web-pages, e.g., HTML files, etc.

Create a New AdvoCATE Project





To create a new project, either

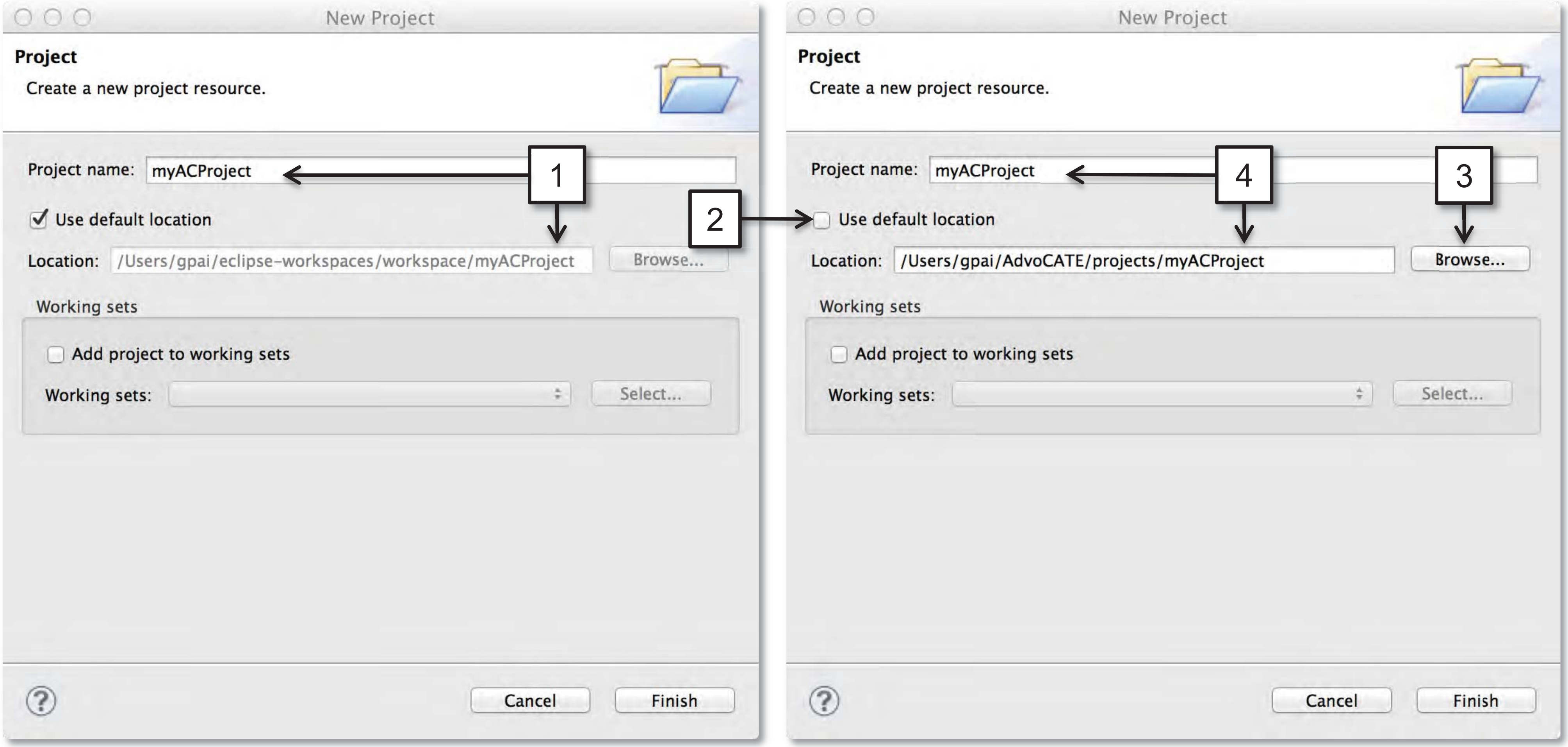
1. Select File > New > Advocate Project Or
2. Select the *New* icon

Note:

The same steps can be used also to create new

A. Argument structures
B. Argument patterns
C. Modules
D. Contract modules

Create a New AdvoCATE Project



New Project

Project

Create a new project resource.

Project name: myACProject

☑ Use default location

Location: /Users/gpai/eclipse-workspaces/workspace/myACProject

Working sets

□ Add project to working sets

Working sets:

New Project

Project

Create a new project resource.

Project name: myACProject

□ Use default location

Location: /Users/gpai/AdvoCATE/projects/myACProject

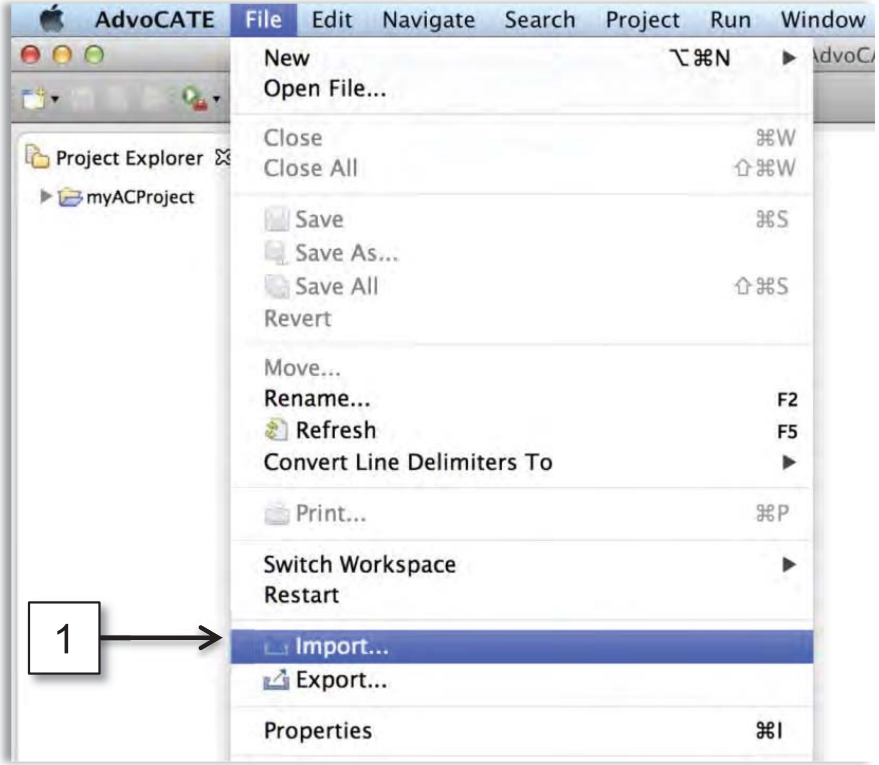
Working sets

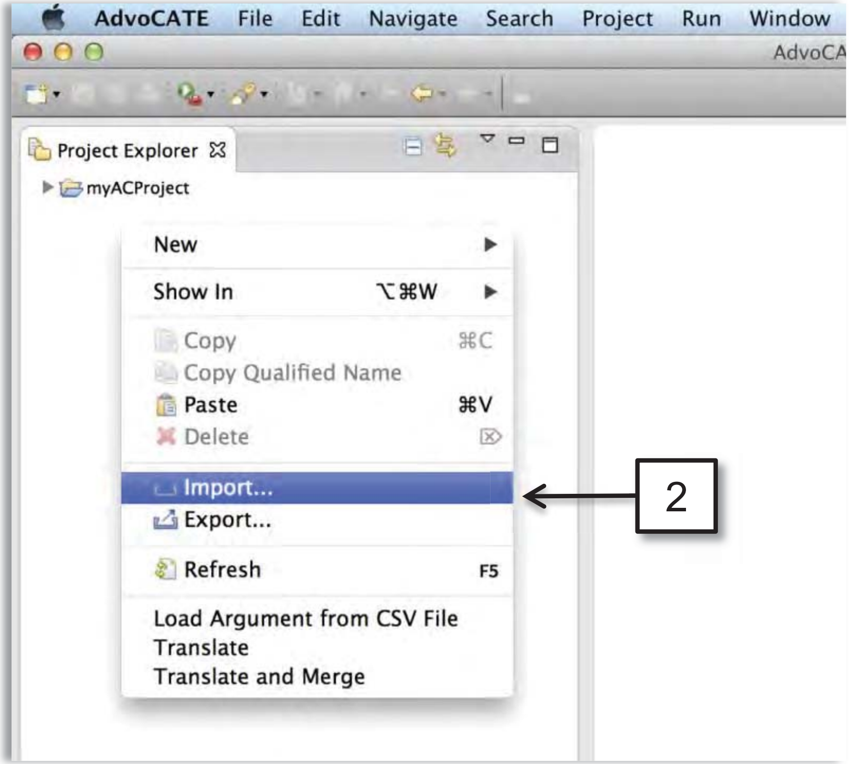
□ Add project to working sets

Working sets:

1. By default the location of a new project is in the workspace.
2. We recommend selecting a different location by unchecking the default location option as workspaces may need to be periodically deleted
3. Specify a location using the *Browse* option
4. Then ensure that folder in which the project will be located has the **same name** as the project

Import an Existing AdvoCATE Project

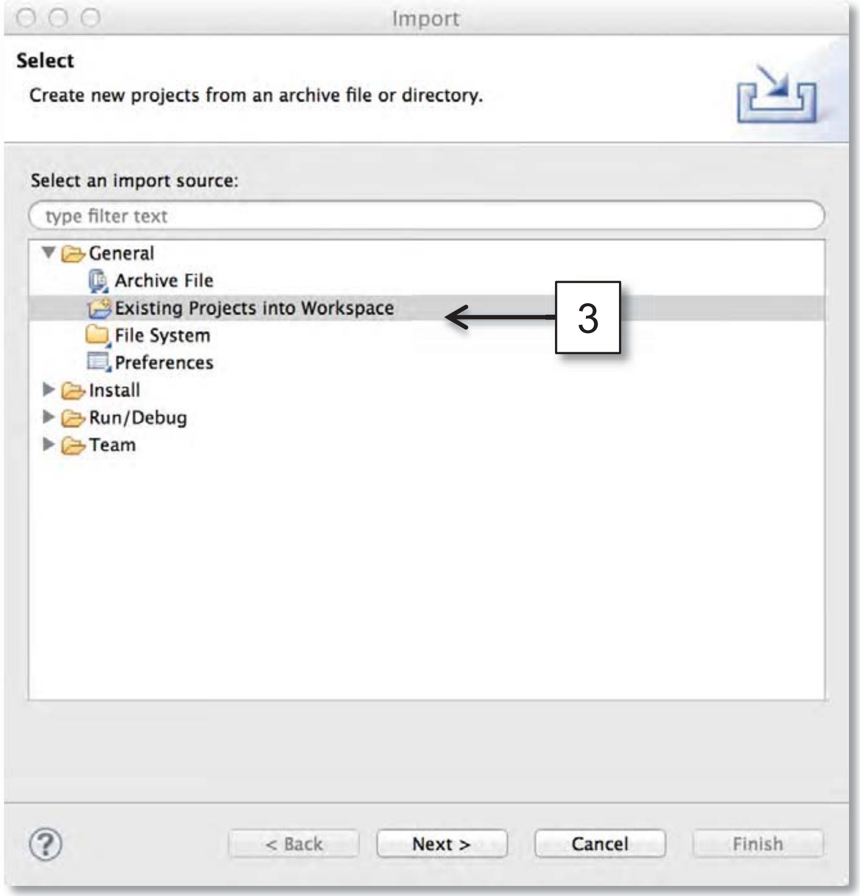




To import an existing project, either

1. Select File > Import OR
2. Right click the mouse in the Project Explorer and select *Import*

Import an Existing AdvoCATE Project

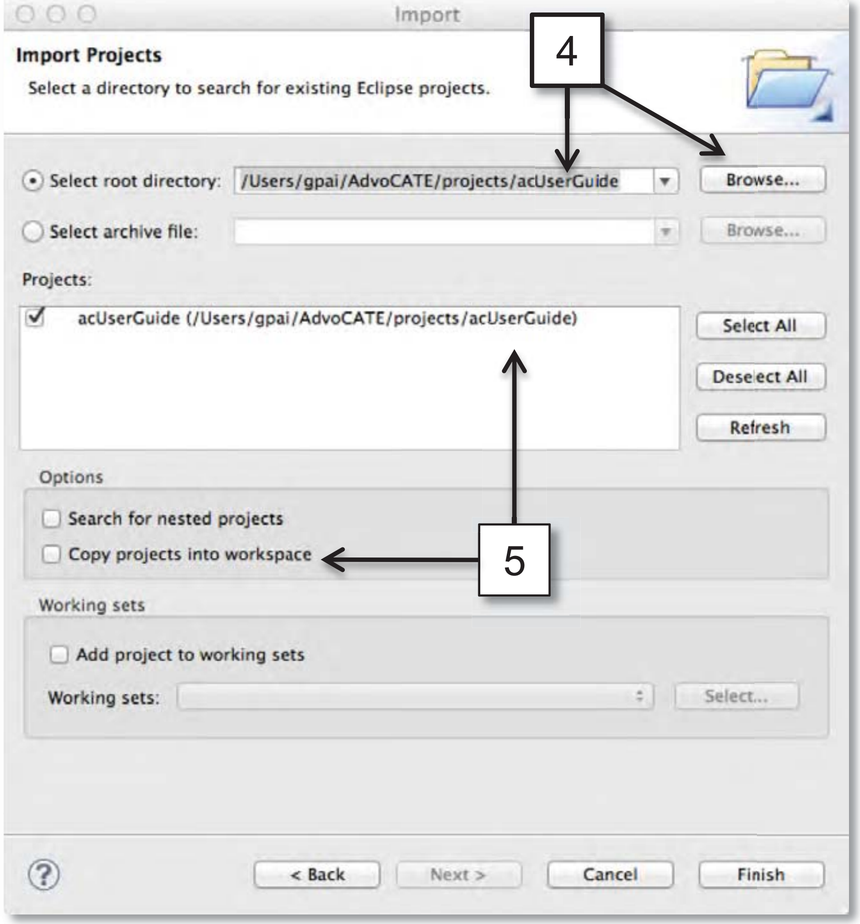


Import

Select

Create new projects from an archive file or directory.

Select an import source:



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /Users/gpai/AdvoCATE/projects/acUserGuide

○ Select archive file:

Projects:

☑ acUserGuide (/Users/gpai/AdvoCATE/projects/acUserGuide)

Options

☐ Search for nested projects

☐ Copy projects into workspace

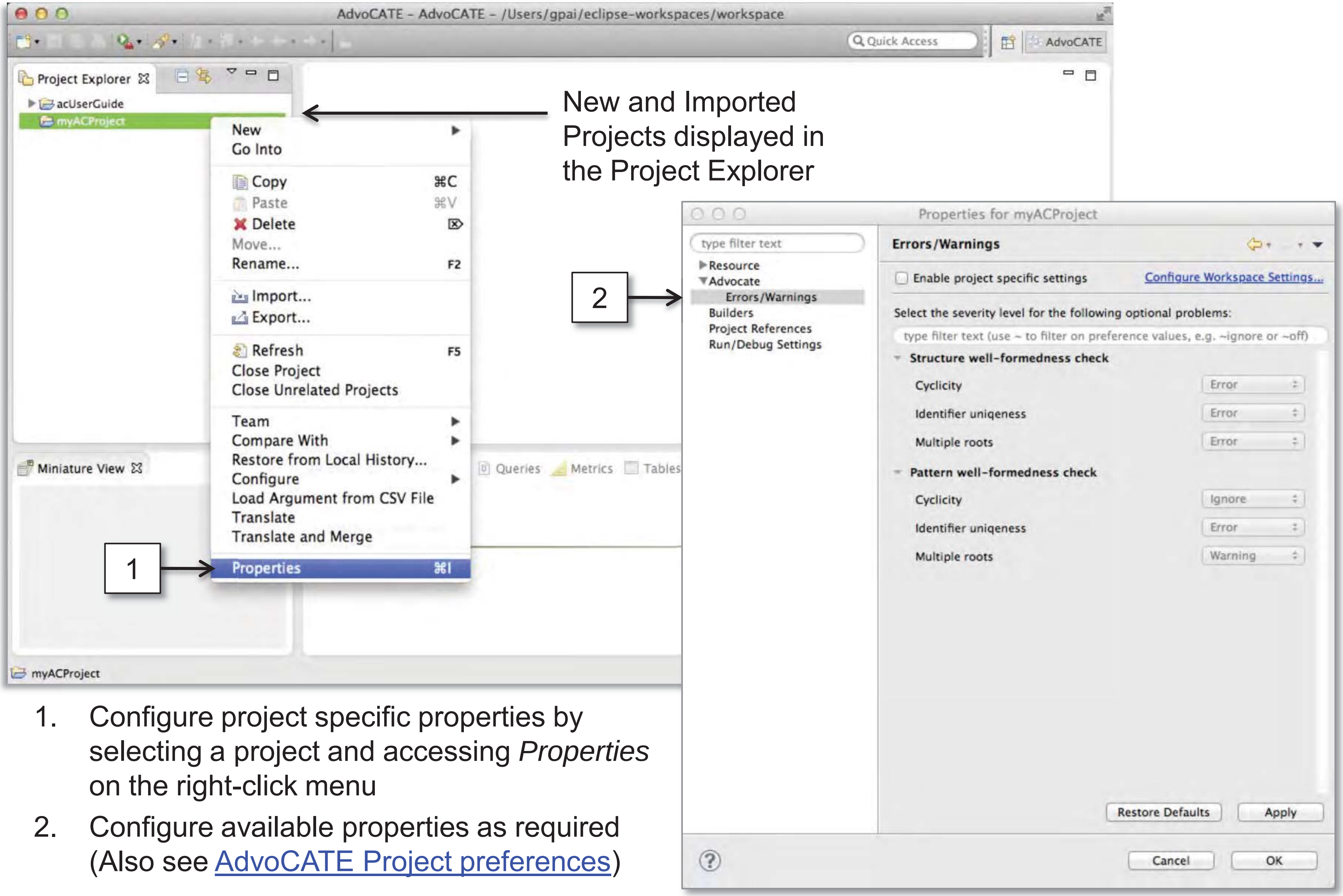
Working sets

☐ Add project to working sets

Working sets:

3. Select *General* > *Existing Projects Into Workspace*
4. Specify a path by selecting the *Browse* button
5. Select the project(s) to import. We recommend not copying the projects into the workspace as workspaces may need to be periodically deleted

AdvoCATE Projects



New and Imported Projects displayed in the Project Explorer

Errors/Warnings

☐ Enable project specific settings   Configure Workspace Settings...

Select the severity level for the following optional problems:

type filter text (use ~ to filter on preference values, e.g. ~ignore or ~off)

Structure well-formedness check

Cyclicity	Error
Identifier uniqeness	Error
Multiple roots	Error

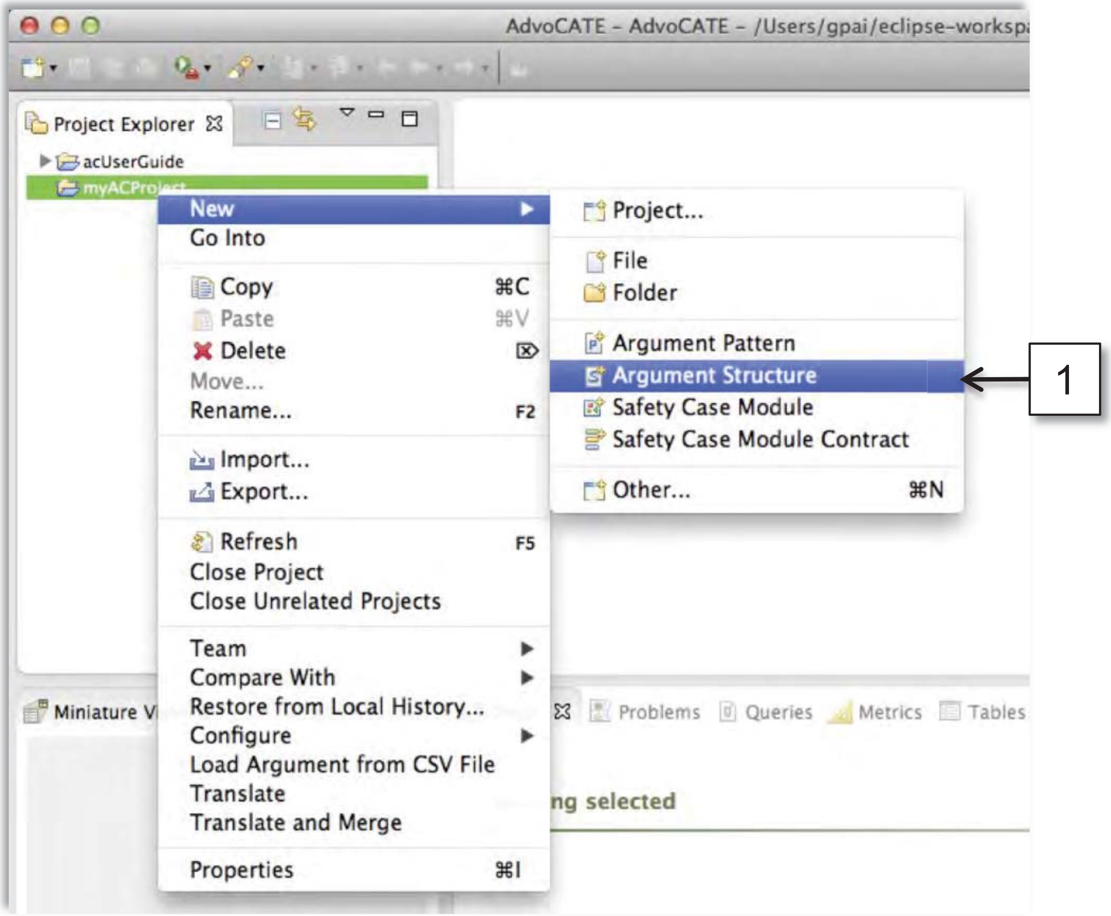
Pattern well-formedness check

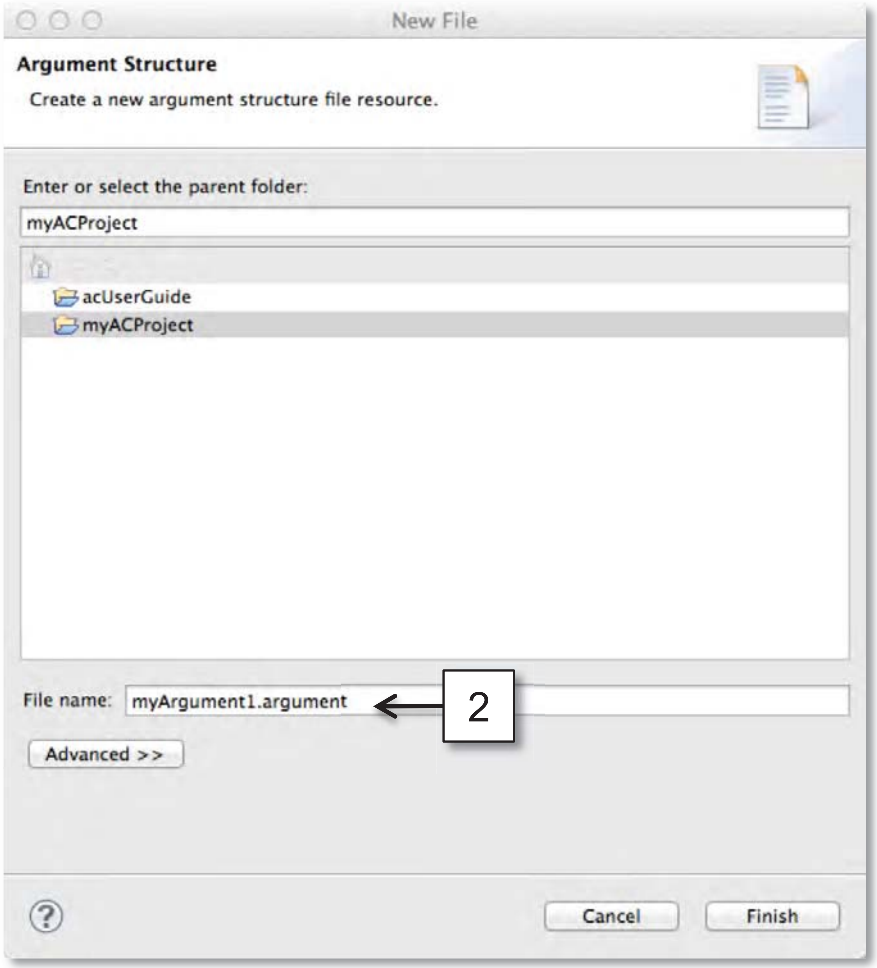
Cyclicity	Ignore
Identifier uniqeness	Error
Multiple roots	Warning

1. Configure project specific properties by selecting a project and accessing *Properties* on the right-click menu
2. Configure available properties as required (Also see AdvoCATE Project preferences)

4. Argument Structures and Patterns

Create a New Argument Structure





Argument Structure

Create a new argument structure file resource.

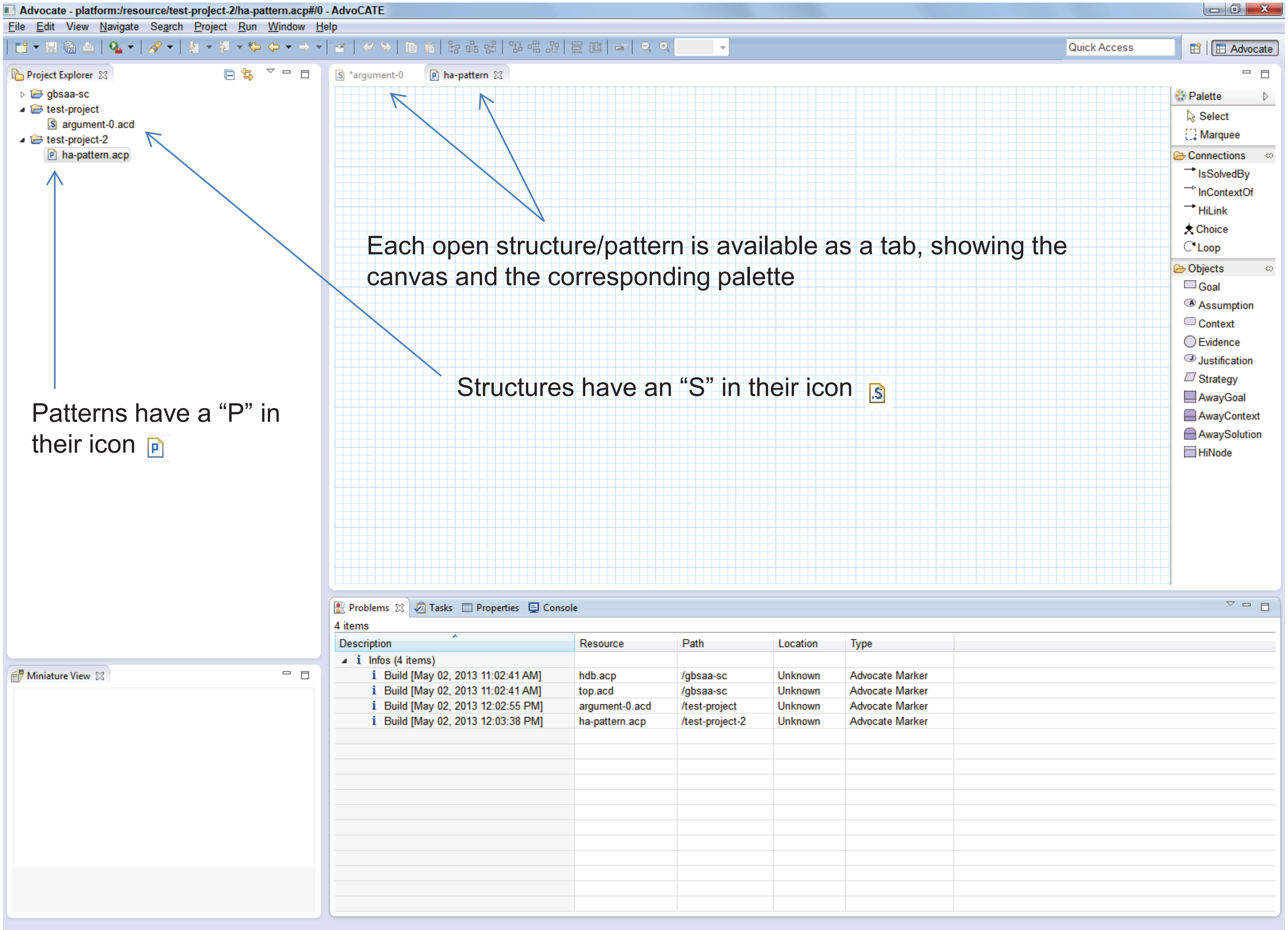
Enter or select the parent folder: myACProject

File name: myArgument1.argument

1. Select a project and access the *New > Argument Structure* option in the right-click menu*
2. Specify a file name; the file extension is automatically added

* A new argument structure can also be created using the *New* icon, or the *File* menu (See Create a New AdvoCATE Project)

New Argument Structure/Pattern



Each open structure/pattern is available as a tab, showing the canvas and the corresponding palette

Structures have an “S” in their icon S

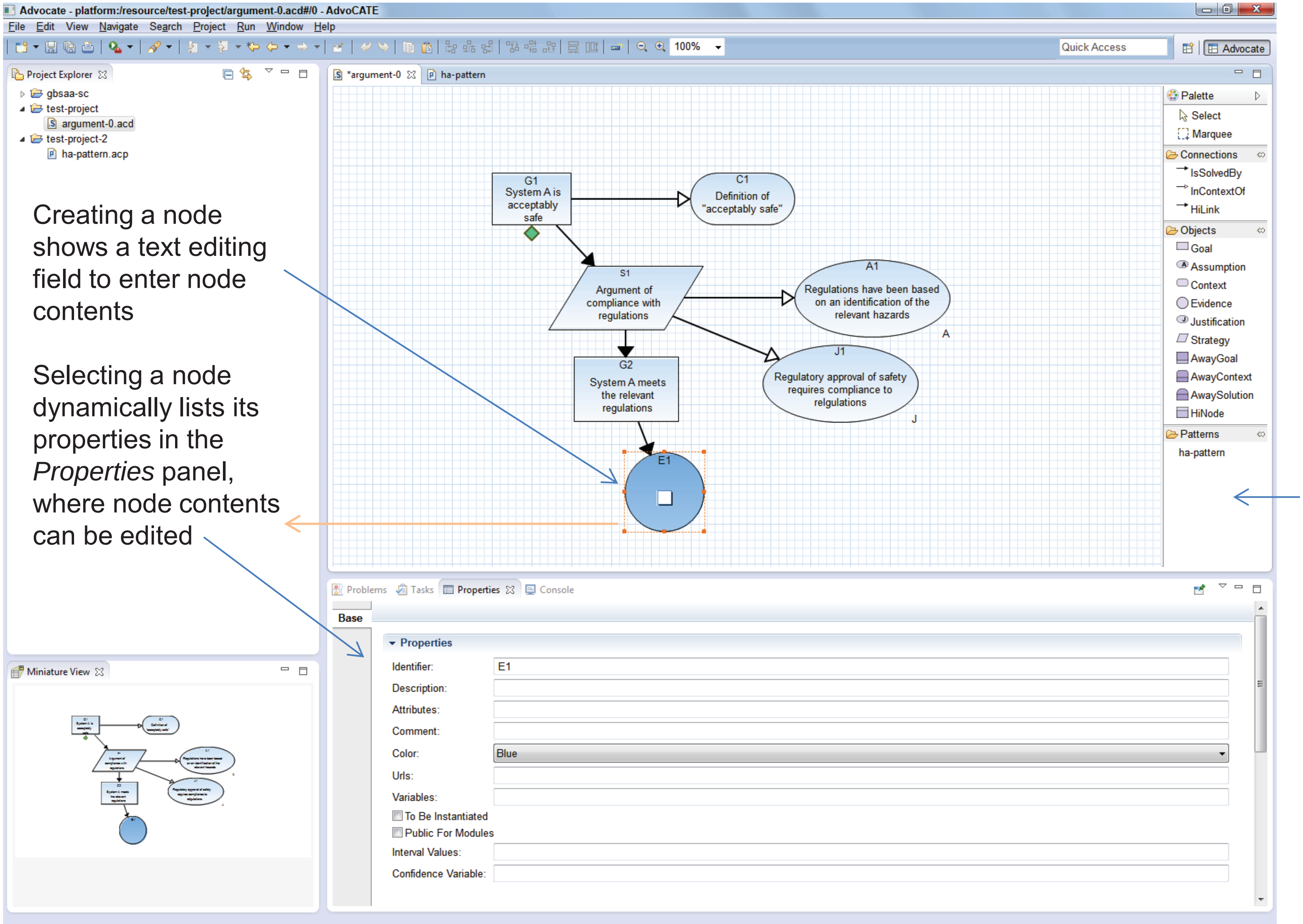
Patterns have a “P” in their icon P

4 items

Description	Resource	Path	Location	Type
Infos (4 items)				
Build [May 02, 2013 11:02:41 AM]	hdb.acp	/gbsaa-sc	Unknown	Advocate Marker
Build [May 02, 2013 11:02:41 AM]	top.acd	/gbsaa-sc	Unknown	Advocate Marker
Build [May 02, 2013 12:02:55 PM]	argument-0.acd	/test-project	Unknown	Advocate Marker
Build [May 02, 2013 12:03:38 PM]	ha-pattern.acp	/test-project-2	Unknown	Advocate Marker

New Argument Structure

Drag and drop nodes (objects) and links (connections) from the palette onto the canvas to create an argument structure



Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Project Run Window Help
100%
Quick Access
Advocate
Project Explorer
gbsaa-sc
test-project
argument-0.acd
test-project-2
ha-pattern.acp
*argument-0
ha-pattern
G1
System A is acceptably safe
C1
Definition of "acceptably safe"
S1
Argument of compliance with regulations
A1
Regulations have been based on an identification of the relevant hazards
A
J1
Regulatory approval of safety requires compliance to relgulations
J
G2
System A meets the relevant regulations
E1
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ha-pattern
Problems
Tasks
Properties
Console
Base
Properties
Miniature View

Identifier: E1

Description:

Attributes:

Comment:

Color: Blue

Urls:

Variables:

□ To Be Instantiated

□ Public For Modules

Interval Values:

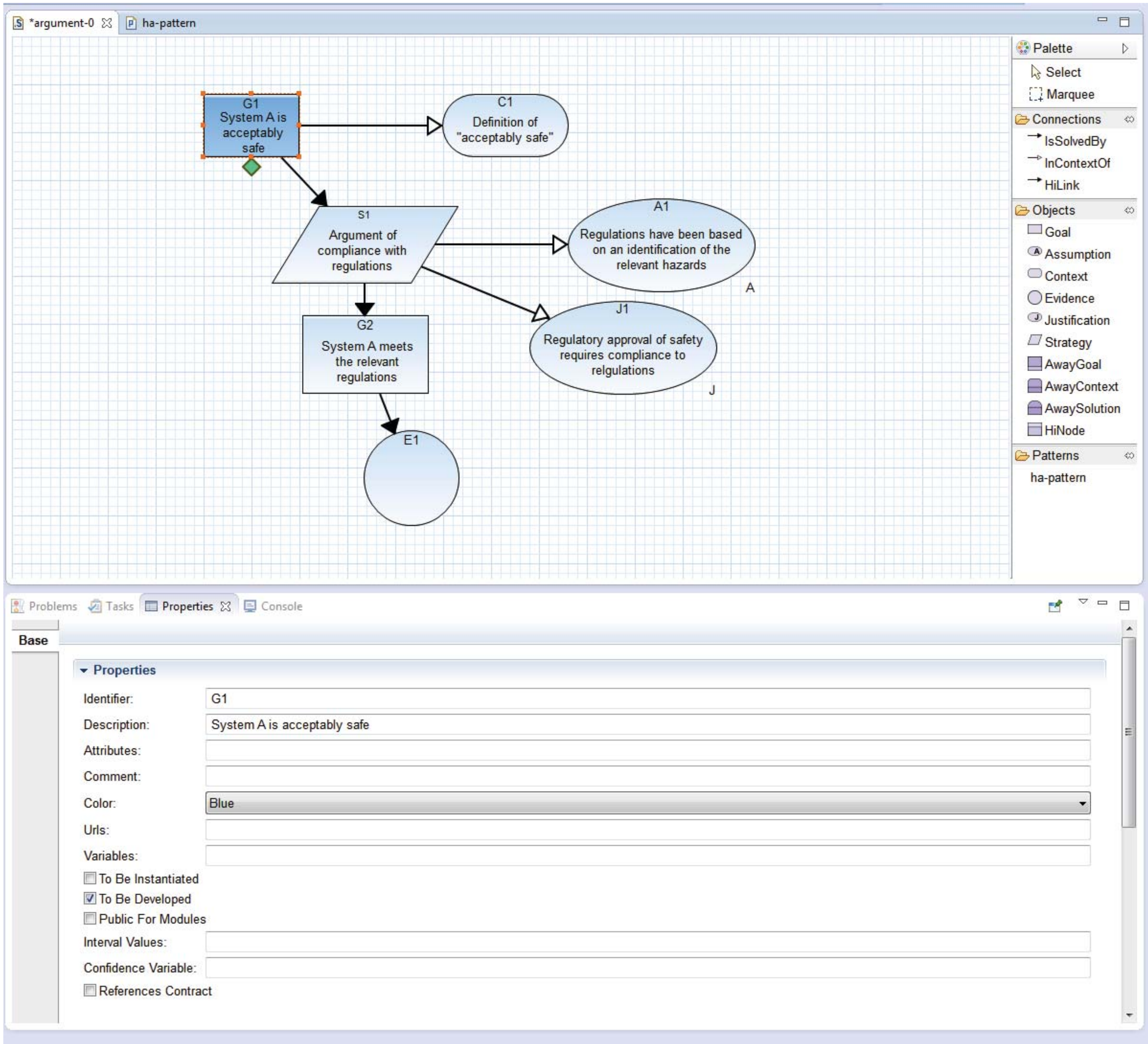
Confidence Variable:

Creating a node shows a text editing field to enter node contents

Selecting a node dynamically lists its properties in the *Properties* panel, where node contents can be edited

A pattern library lists all patterns in the current and referenced projects

New Argument Structure


*argument-0
ha-pattern
G1
System A is acceptably safe
C1
Definition of "acceptably safe"
S1
Argument of compliance with regulations
A1
Regulations have been based on an identification of the relevant hazards
A
J1
Regulatory approval of safety requires compliance to relguations
J
G2
System A meets the relevant regulations
E1
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ha-pattern
Problems
Tasks
Properties
Console
Base
Properties

Identifier: G1

Description: System A is acceptably safe

Attributes:

Comment:

Color: Blue

Urls:

Variables:

□ To Be Instantiated

☑ To Be Developed

□ Public For Modules

Interval Values:

Confidence Variable:

□ References Contract

Each node has:

- An auto-generated, but user-editable, *Identifier*
- A *Description* field to enter node contents
- A *Metadata* field to enter node-specific metadata
- User-specified *Comments*
- A *Hyperlinks* field to add links to Web urls
- Checkboxes to ~~decorate~~ annotate nodes as
  - To be instantiated (Only relevant for patterns)
  - To be developed
  - Public (Only relevant for Goal, Context and Evidence nodes to make them publicly accessible in external module)
  - To Be Supported By Contract
- An *Association* dropdown menu and field to indicate whether a node is associated with another argument

~~Specific nodes have specific additional fields.~~

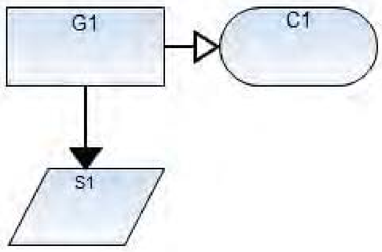
- ~~e.g., “Public for modules”, in Goal, Context and Evidence nodes to make them publicly accessible in external modules~~

Keyboard Shortcuts

Keyboard shortcuts are an alternative to using the palette for rapidly creating simple argument structures/patterns.

To use keyboard shortcuts to create an argument structure

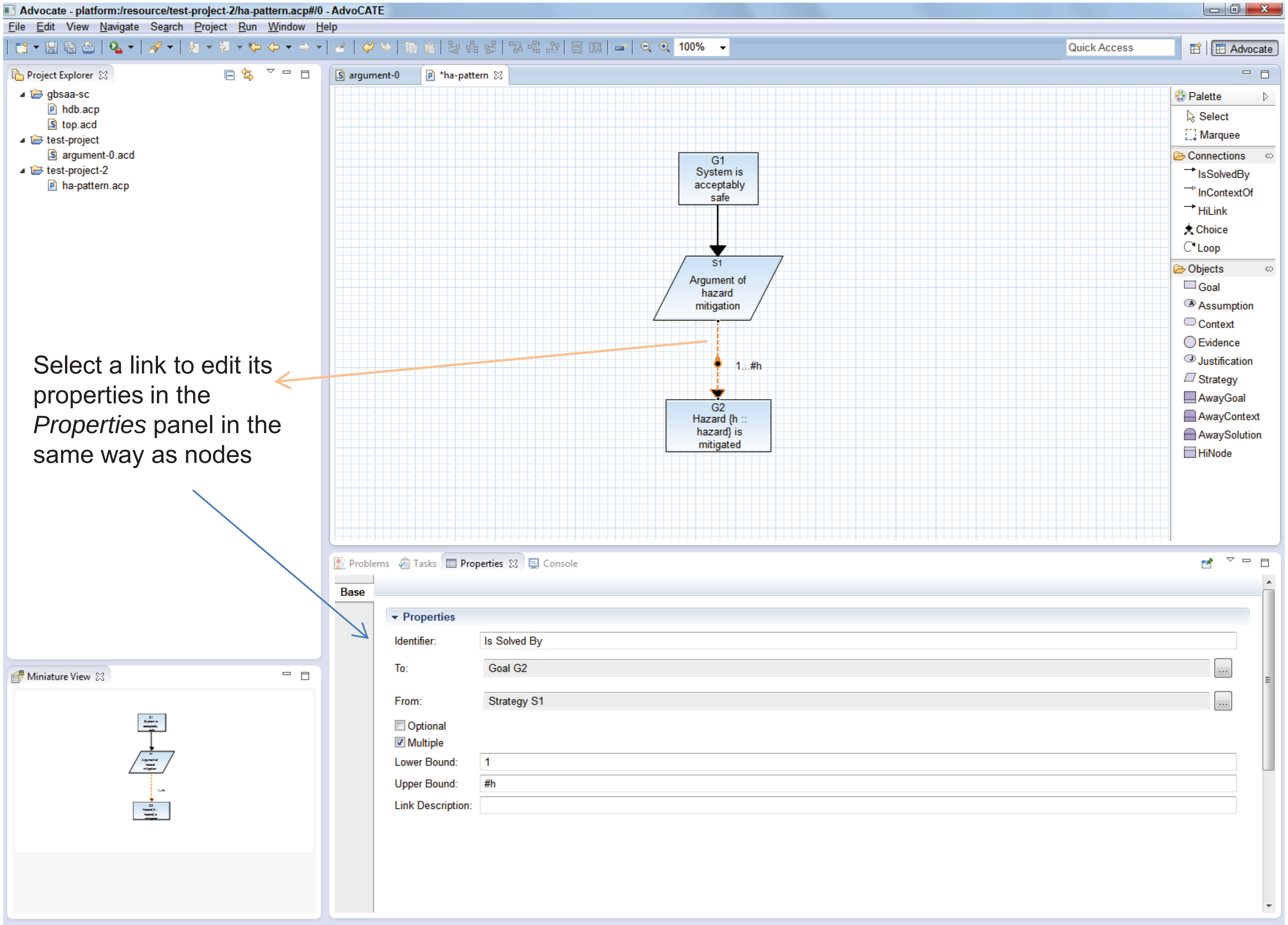
1. Make the canvas active by selecting it
2. Select the appropriate keystroke to create a node
3. By default, a text box appears to specify node contents.
4. Select the next appropriate keystroke to create a new node.
5. AdvoCATE automatically links the two nodes with the appropriate link.
6. Use the arrow keys to select the node to which the next new node is to be attached
7. For example, G S ↑ C creates the following structure



Node	Keystroke
Assumption	Alt + A
Context	Alt + C
Evidence (Solution)	Alt + E
Goal	Alt + G
Justification	Alt + J
Strategy	Alt + S
Away Goal	Alt + ⇧ Shift + G
Away Context	Alt + ⇧ Shift + C
Away Solution	Alt + ⇧ Shift + E
Module Reference	Alt + M
Module Contract Reference	Alt + ⇧ Shift + M
Navigation Between Nodes	↑ ↓ → ←
Search	Ctrl + H
Select all nodes	Ctrl + A

Create a New Argument Pattern

Drag and drop objects (nodes) or connections (links) to create a pattern in the same way as structures



Properties

Identifier: Is Solved By

To: Goal G2

From: Strategy S1

☐ Optional

☑ Multiple

Lower Bound: 1

Upper Bound: #h

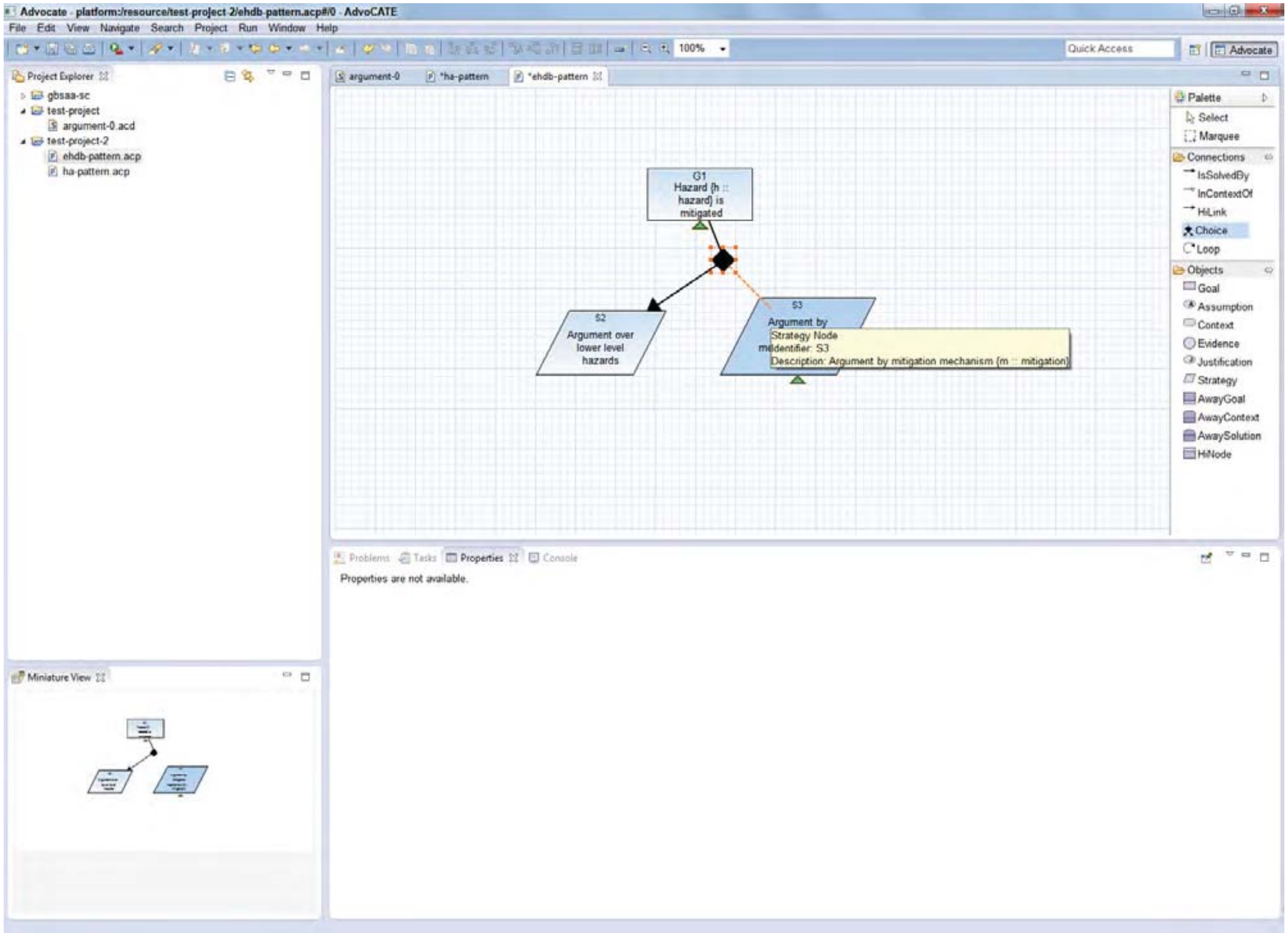
Link Description:

Select a link to edit its properties in the *Properties* panel in the same way as nodes

Argument Patterns – Choices

To create a choice:

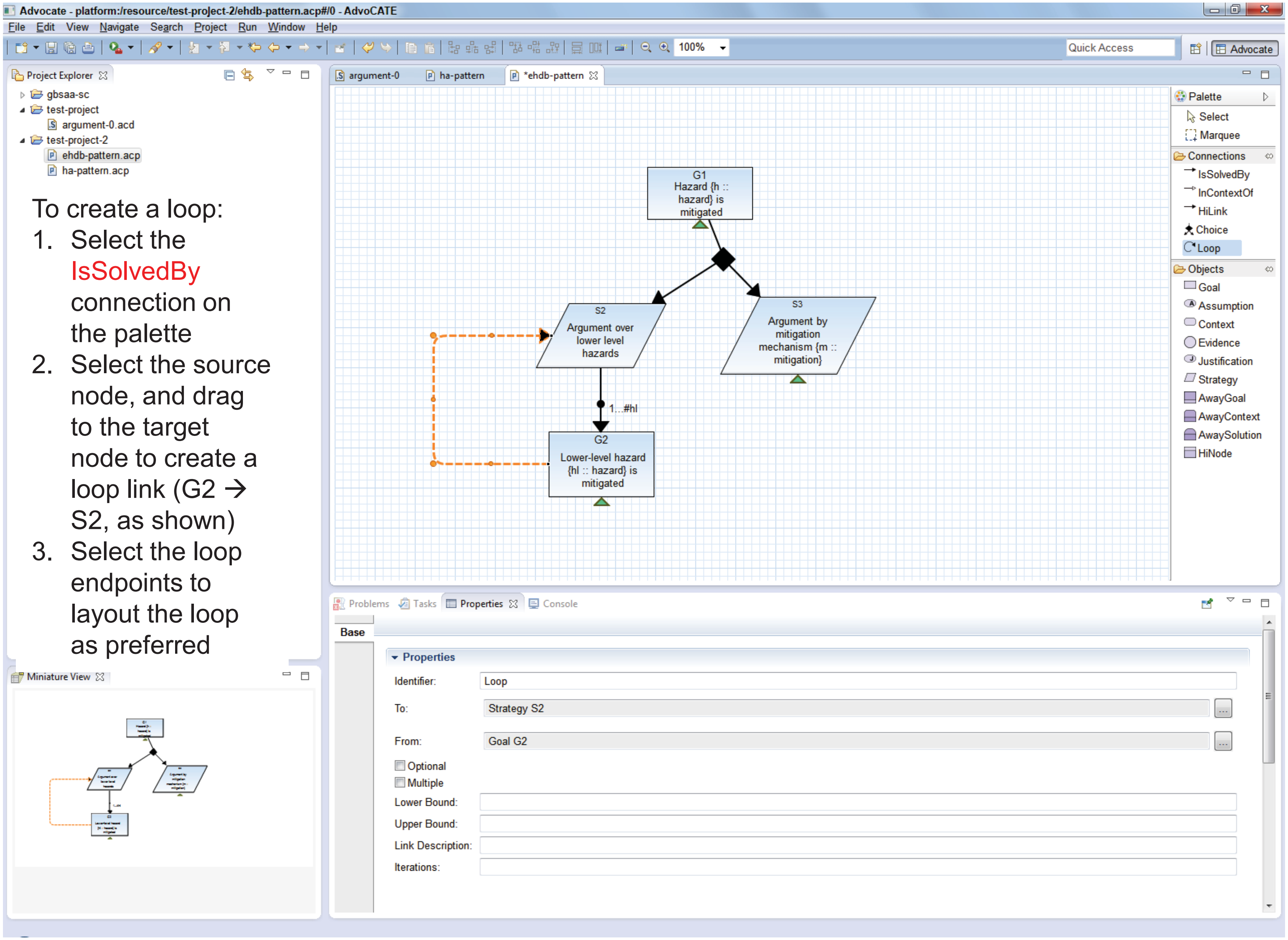
1. Select the choice connection on the palette
2. Then select the source node, and drag to the target node to create a choice link (e.g., G1 → S2 as shown)
3. Re-select the choice connection on the palette
4. Then select the choice (◆) on the canvas and drag to the second target (e.g., ◆ → S3 as shown), and so on



Strategy Node
Identifier: S3
Description: Argument by mitigation mechanism {m :: mitigation}

Properties are not available.

Argument Patterns – Loops



To create a loop:

1. Select the IsSolvedBy connection on the palette
2. Select the source node, and drag to the target node to create a loop link (G2 → S2, as shown)
3. Select the loop endpoints to layout the loop as preferred

Properties

Identifier: Loop

To: Strategy S2

From: Goal G2

☐ Optional

☐ Multiple

Lower Bound:

Upper Bound:

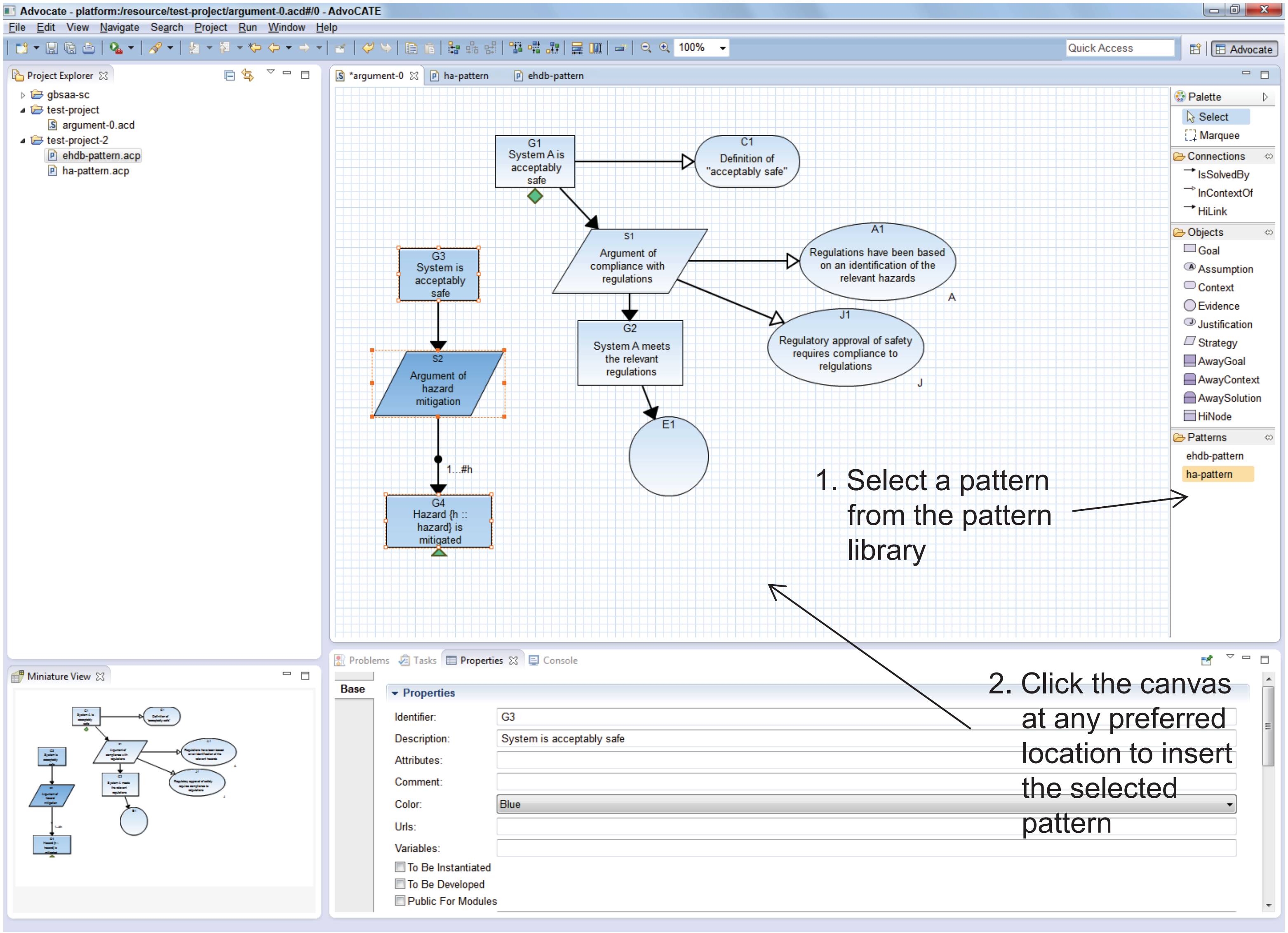
Link Description:

Iterations:

Inserting Patterns from the Pattern Library

Patterns can be added to the canvas either through instantiation (to be discussed later) or by inserting them directly from the pattern library.

AdvoCATE automatically ensures unique identifiers by auto-generating them.


Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Project Run Window Help
100%
Quick Access
Advocate
Project Explorer
gbsaa-sc
test-project
argument-0.acd
test-project-2
ehdb-pattern.acp
ha-pattern.acp
*argument-0
ha-pattern
ehdb-pattern
G1
System A is acceptably safe
C1
Definition of "acceptably safe"
S1
Argument of compliance with regulations
A1
Regulations have been based on an identification of the relevant hazards
A
G3
System is acceptably safe
G2
System A meets the relevant regulations
J1
Regulatory approval of safety requires compliance to regulations
J
S2
Argument of hazard mitigation
E1
1...#h
G4
Hazard {h :: hazard} is mitigated
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ehdb-pattern
ha-pattern
Miniature View
Problems
Tasks
Properties
Console
Base

1. Select a pattern from the pattern library

2. Click the canvas at any preferred location to insert the selected pattern

Properties

Identifier: G3

Description: System is acceptably safe

Attributes:

Comment:

Color: Blue

Urls:

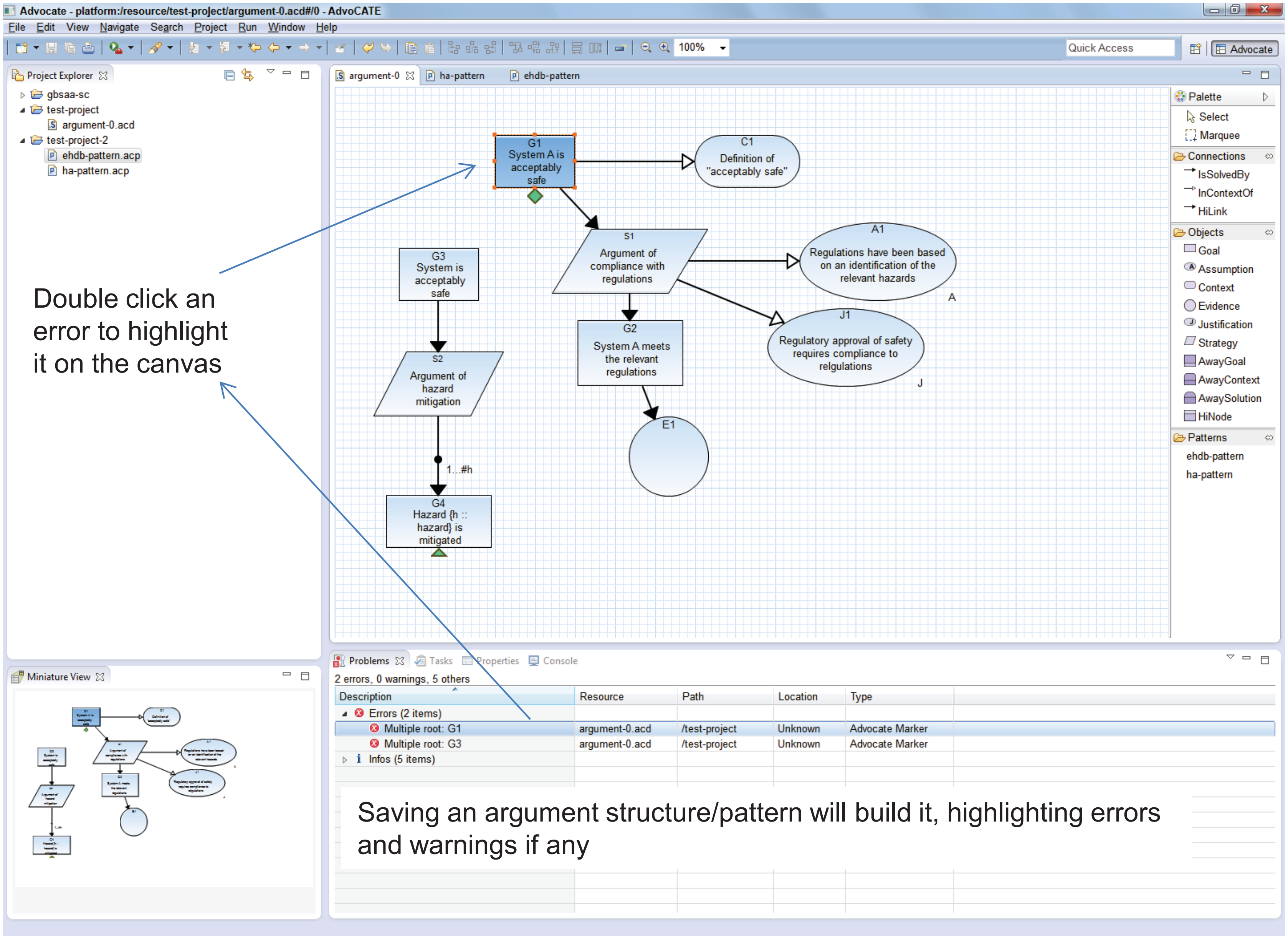
Variables:

□ To Be Instantiated

□ To Be Developed

□ Public For Modules

Checking Well-Formedness


Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Project Run Window Help
100%
Quick Access
Advocate
Project Explorer
gbsaa-sc
test-project
argument-0.acd
test-project-2
ehdb-pattern.acp
ha-pattern.acp
argument-0
ha-pattern
ehdb-pattern
G1
System A is acceptably safe
C1
Definition of "acceptably safe"
S1
Argument of compliance with regulations
A1
Regulations have been based on an identification of the relevant hazards
A
G3
System is acceptably safe
J1
Regulatory approval of safety requires compliance to relgulations
J
G2
System A meets the relevant regulations
S2
Argument of hazard mitigation
E1
1...#h
G4
Hazard {h :: hazard} is mitigated
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ehdb-pattern
ha-pattern
Miniature View
Problems
Tasks
Properties
Console

2 errors, 0 warnings, 5 others

Description	Resource	Path	Location	Type
Errors (2 items)				
Multiple root: G1	argument-0.acd	/test-project	Unknown	Advocate Marker
Multiple root: G3	argument-0.acd	/test-project	Unknown	Advocate Marker
Infos (5 items)				

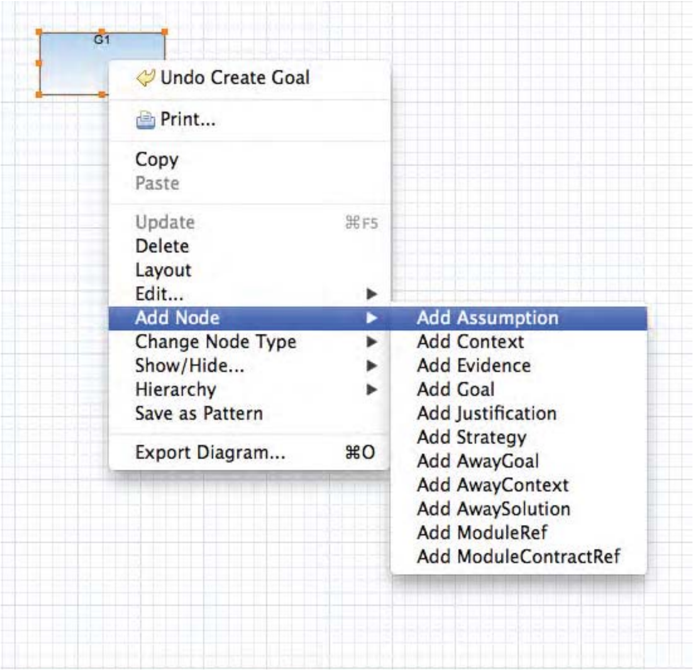
Double click an error to highlight it on the canvas

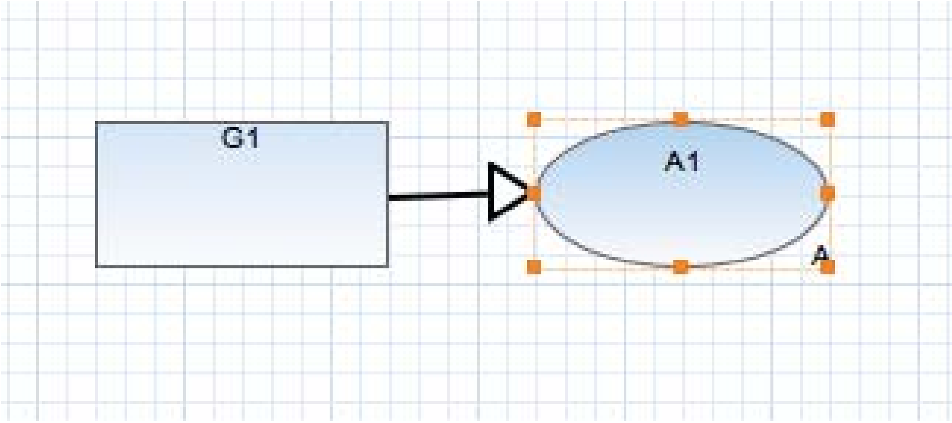
Saving an argument structure/pattern will build it, highlighting errors and warnings if any

Other Ways to Add Nodes

Another way to add a node to a diagram is to:

1. Right-click the node.
2. Select Add Node and then any node choice. For example, Add Node→ Add Assumption will automatically add an Assumption node as well as the appropriate link to the diagram.

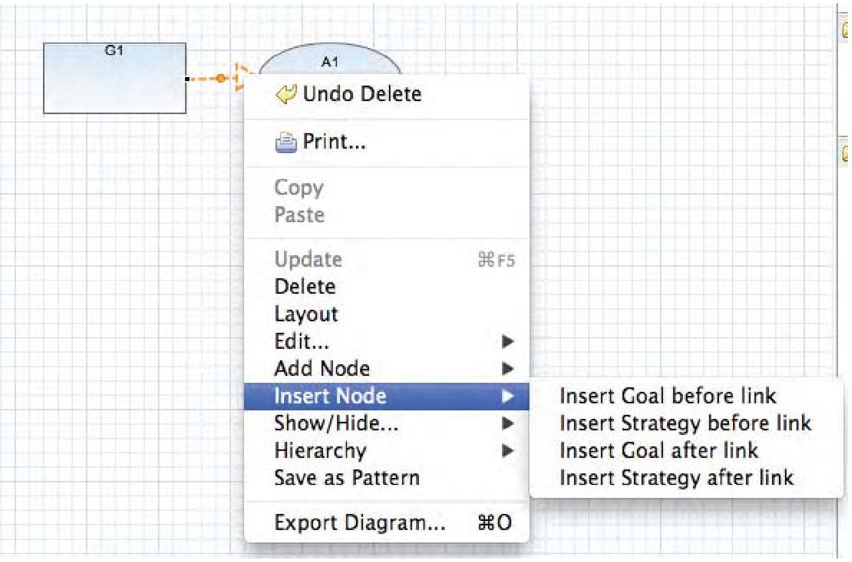


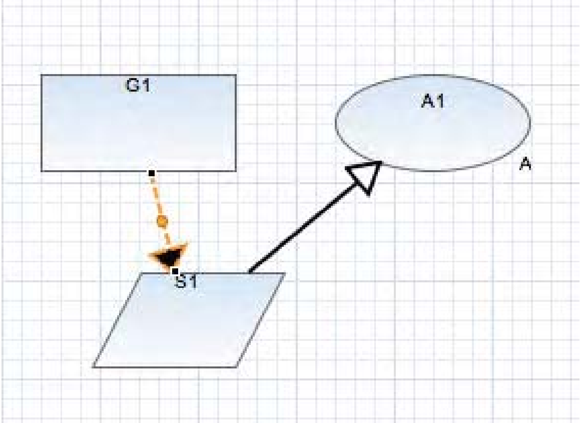


Other Ways to Add Nodes (Cont.)

Another way to add a node to a diagram is to:

1. Right-click a link that is connecting two nodes.
2. Select Insert Node and then the option of Goal or Strategy nodes before or after the link. For example, selecting Insert Node→ Insert Strategy before link will produce the diagram at right.



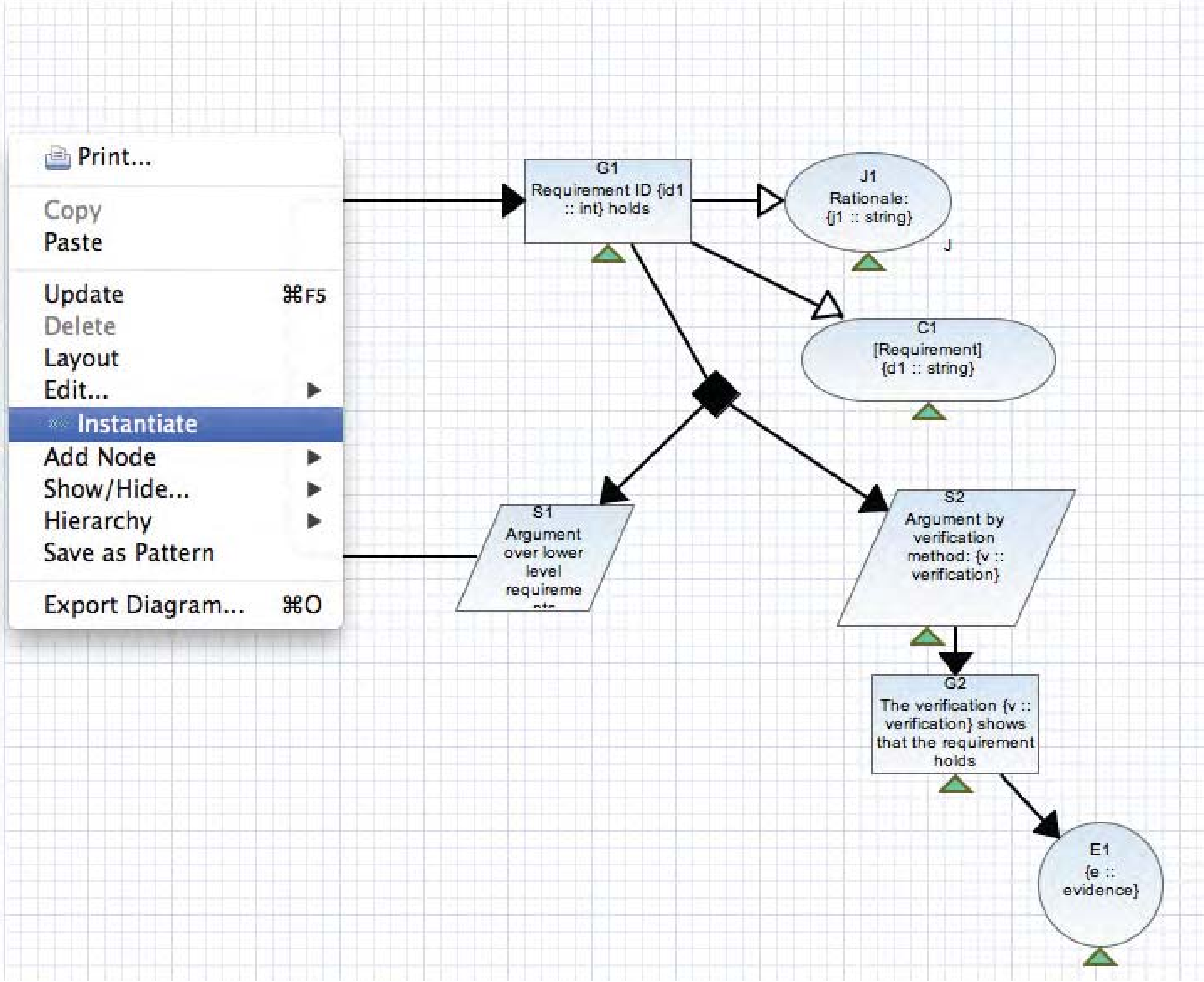


Pattern Instantiation

Instantiating a pattern involves systematically replacing its node parameters with new or different values. The AdvoCATE tool uses a .csv file to specify which parameters are replaced and with what values.

To instantiate a pattern:

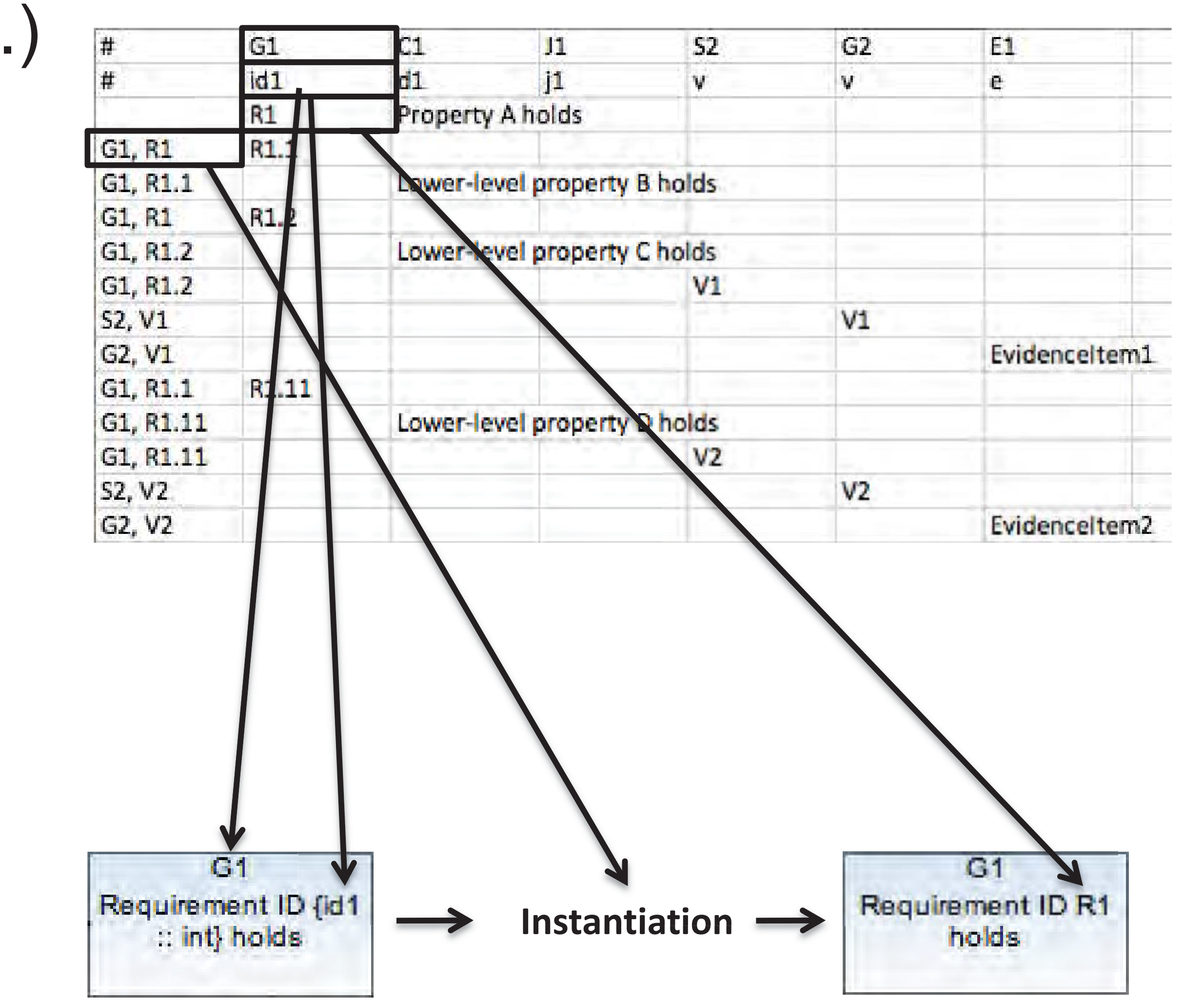
1. Open the pattern.
2. Right-click on the canvas.
3. Select Instantiate.
4. From your local file structure, select .csv file from which to pull values.


Print...
Copy
Paste
Update ⌘F5
Delete
Layout
Edit...
Instantiate
Add Node
Show/Hide...
Hierarchy
Save as Pattern
Export Diagram... ⌘O
G1
Requirement ID {id1 :: int} holds
J1
Rationale: {j1 :: string}
J
C1
[Requirement] {d1 :: string}
S1
Argument over lower level requireme nts
S2
Argument by verification method: {v :: verification}
G2
The verification {v :: verification} shows that the requirement holds
E1
{e :: evidence}

Pattern Instantiation (cont.)

For example, in the .csv file shown here:

- Row 1 specifies the node identifier for nodes in the pattern that have parameters
- Row 2 specifies the parameter identifiers in the corresponding nodes as {variable:type}
- Column 1 shows how the specified node will be instantiated—i.e. it specifies the values of the parameters and the locations in the instance where each node is to be joined



#	G1	C1	J1	S2	G2	E1
#	id1	d1	j1	v	v	e
	R1	Property A holds				
G1, R1	R1.1					
G1, R1.1		Lower-level property B holds				
G1, R1	R1.2					
G1, R1.2		Lower-level property C holds				
G1, R1.2				V1		
S2, V1					V1	
G2, V1						EvidenceItem1
G1, R1.1	R1.11					
G1, R1.11		Lower-level property D holds				
G1, R1.11				V2		
S2, V2					V2	
G2, V2						EvidenceItem2

Pattern before instantiation

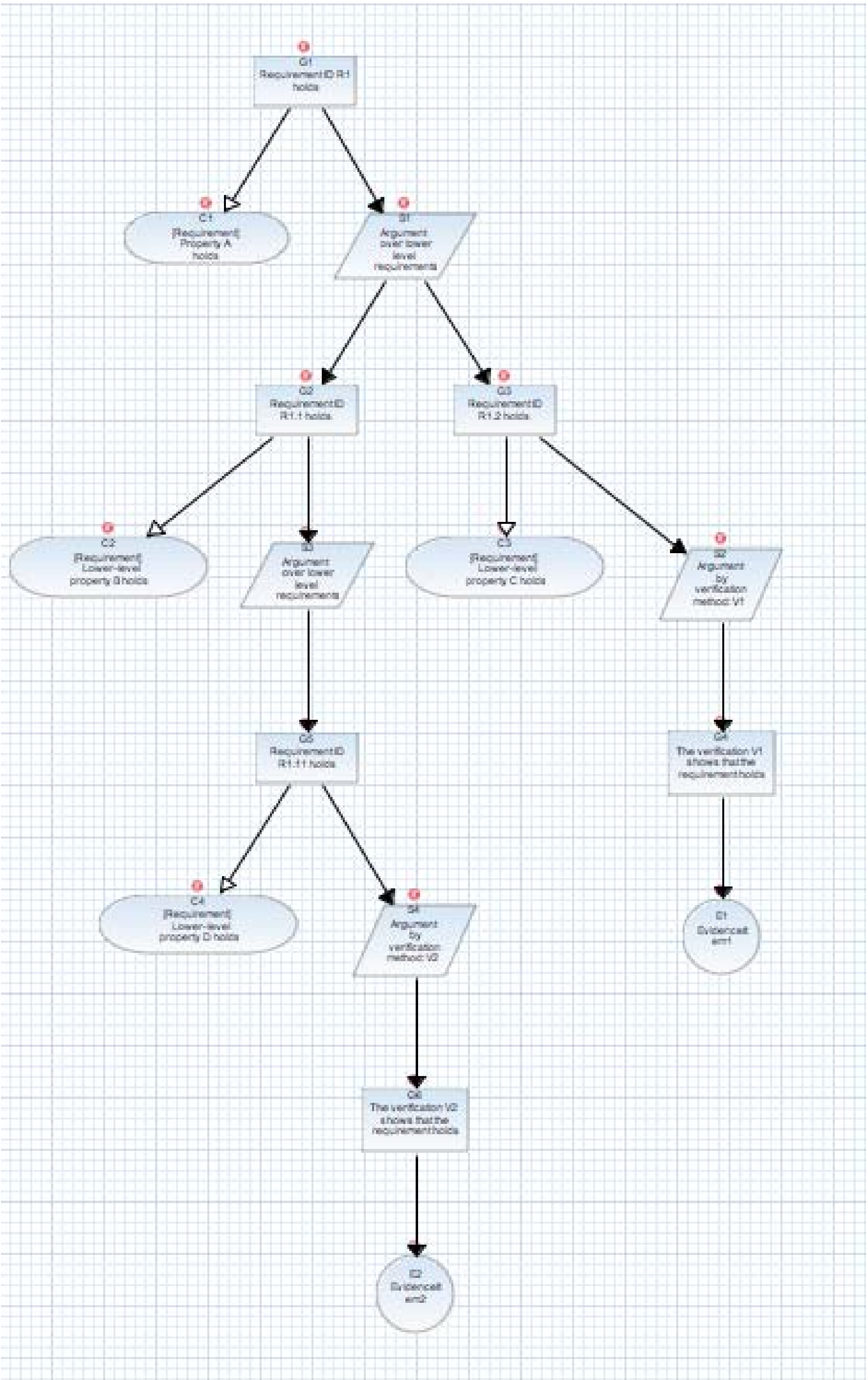
Pattern after instantiation

Pattern Instantiation (cont.)

AdvoCATE will automatically open up a new tab with the instantiated pattern.

The instantiated pattern will also appear in the project menu on the left as XXX-instance-1.argument (where XXX is the name of the original pattern).

The tool features a default auto-layout, as shown, but can be manually modified.


G1
Requirement ID R1 holds
C1
[Requirement] Property A holds
S1
Argument over lower level requirements
G2
Requirement ID R1.1 holds
G3
Requirement ID R1.2 holds
C2
[Requirement] Lower-level property B holds
S3
Argument over lower level requirements
C3
[Requirement] Lower-level property C holds
S2
Argument by verification method V1
G5
Requirement ID R1.11 holds
G4
The verification V1 shows that the requirement holds
C4
[Requirement] Lower-level property D holds
S4
Argument by verification method V2
E1
Evidenceitem1
G6
The verification V2 shows that the requirement holds
E2
Evidenceitem2

Editing Metadata

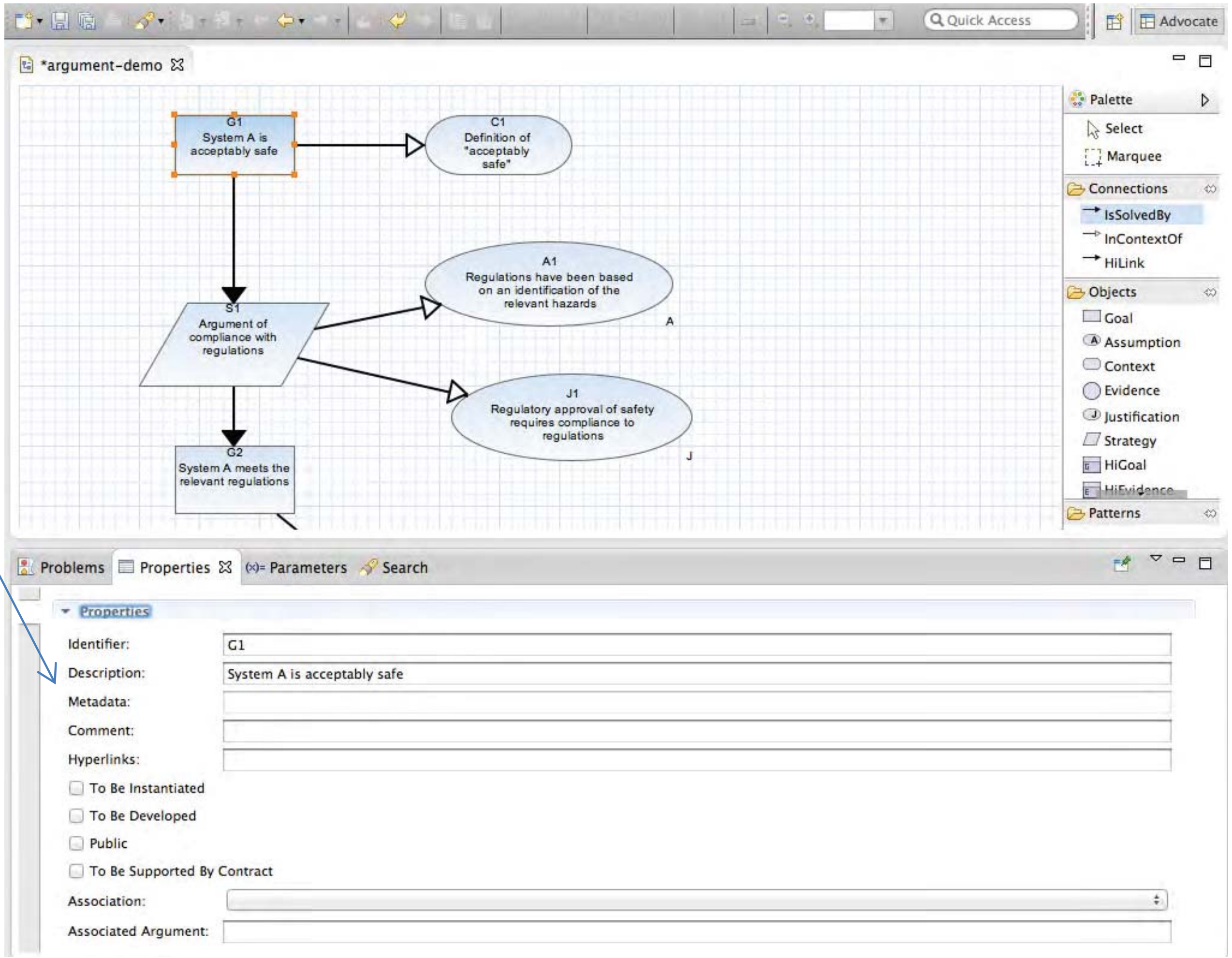
Selecting a node dynamically lists its properties in the *Properties* panel, where node content—like its metadata—can be edited.

To add contents to the metadata field in the node's Properties panel, either:

- Create and add attributes to the node

OR

- Add existing attributes to the node


Quick Access
Advocate
*argument-demo
G1
System A is acceptably safe
C1
Definition of "acceptably safe"
A1
Regulations have been based on an identification of the relevant hazards
A
S1
Argument of compliance with regulations
J1
Regulatory approval of safety requires compliance to regulations
J
G2
System A meets the relevant regulations
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
HiGoal
HiEvidence
Patterns
Problems
Properties
Parameters
Search

Properties

Identifier: G1

Description: System A is acceptably safe

Metadata:

Comment:

Hyperlinks:

☐ To Be Instantiated

☐ To Be Developed

☐ Public

☐ To Be Supported By Contract

Association:

Associated Argument:

Editing Metadata—Creating Attributes and Parameters

An attribute must already exist for it to be added to a node’s metadata.

To create an attribute and its parameters to add it to a specific node,

1. Follow the same initial steps as for editing an attribute. Right-click the node. Select Edit and then Edit Attributes. In the node’s Edit Attributes box, click Add.
2. If the attribute to be added is not listed in the Attribute Selection box, it must be created. Click the Attributes link near the top to open the Preferences box.
3. In the Preferences box, click Add to create an attribute.
4. Name the attribute. Choose its parameter types by clicking Add and individually selecting parameters from the list in the Type Selection box . Click Add to add each parameter to the attribute. Click OK to add the attribute to the Attributes list in the Preferences box.
5. In the Preferences box, click Apply and OK. The newly created attribute should appear in the Attributes Selection box.
6. Select the attribute to add, click OK, and in the Edit Attributes box, select the attribute parameters and click OK.
7. Click OK in the node’s Edit Attributes box. The new attribute should appear in the node’s metadata.

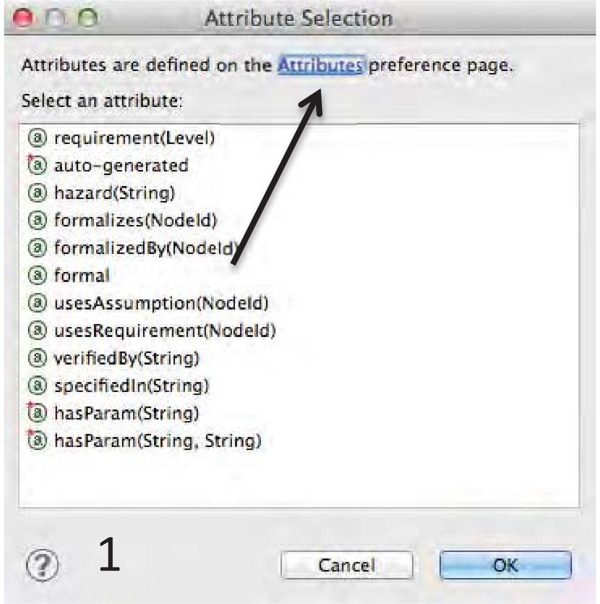
See the following example for creating an attribute and determining its parameters.

Example: Editing Metadata—Creating Attributes and Parameters

An attribute must already exist for it to be added to a node's metadata.

To create an attribute and its parameters to add it to a specific node, first select the node and go to Edit Attributes. In this case, risk does not appear in the Attributes Selection list, so it must be created.

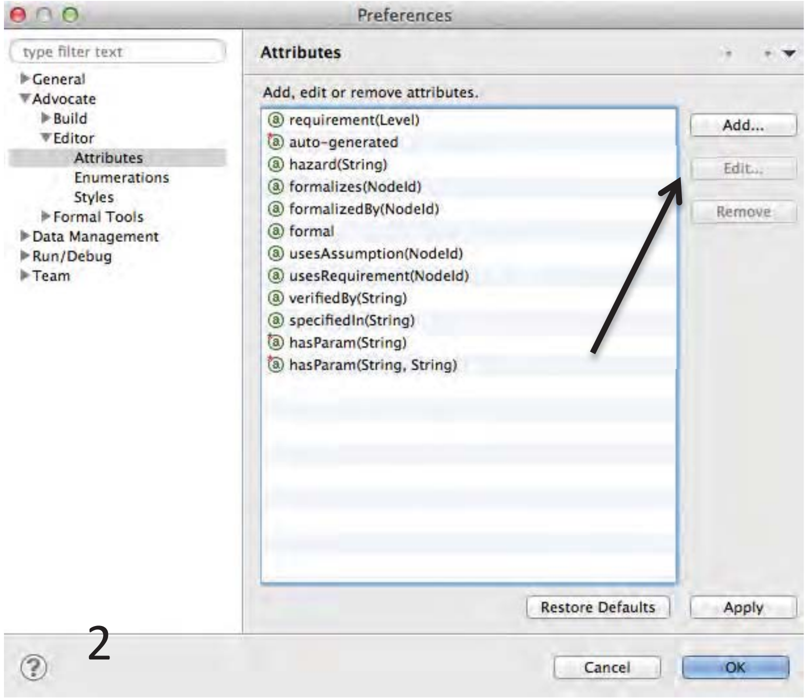
1. Select the Attributes link in the in Attribute Selection box. This opens the Preferences box.
2. In the Preferences box, click Add.


Attribute Selection

Attributes are defined on the Attributes preference page.

Select an attribute:

- requirement(Level)
- auto-generated
- hazard(String)
- formalizes(NodeId)
- formalizedBy(NodeId)
- formal
- usesAssumption(NodeId)
- usesRequirement(NodeId)
- verifiedBy(String)
- specifiedIn(String)
- hasParam(String)
- hasParam(String, String)


Preferences

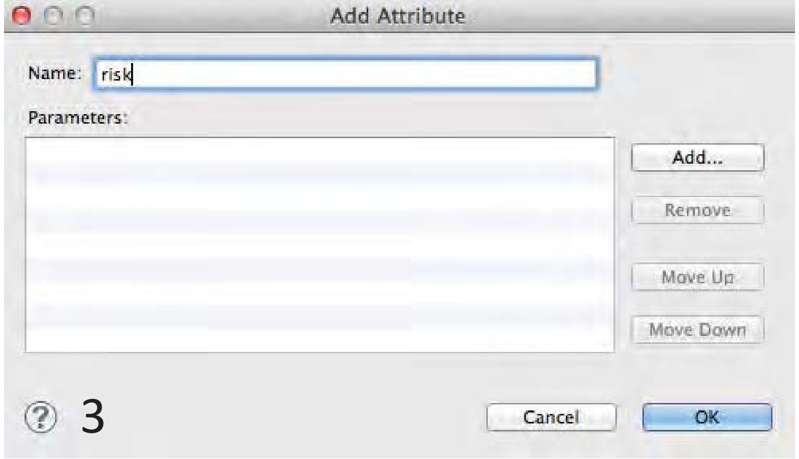
Attributes

Add, edit or remove attributes.

- requirement(Level)
- auto-generated
- hazard(String)
- formalizes(NodeId)
- formalizedBy(NodeId)
- formal
- usesAssumption(NodeId)
- usesRequirement(NodeId)
- verifiedBy(String)
- specifiedIn(String)
- hasParam(String)
- hasParam(String, String)

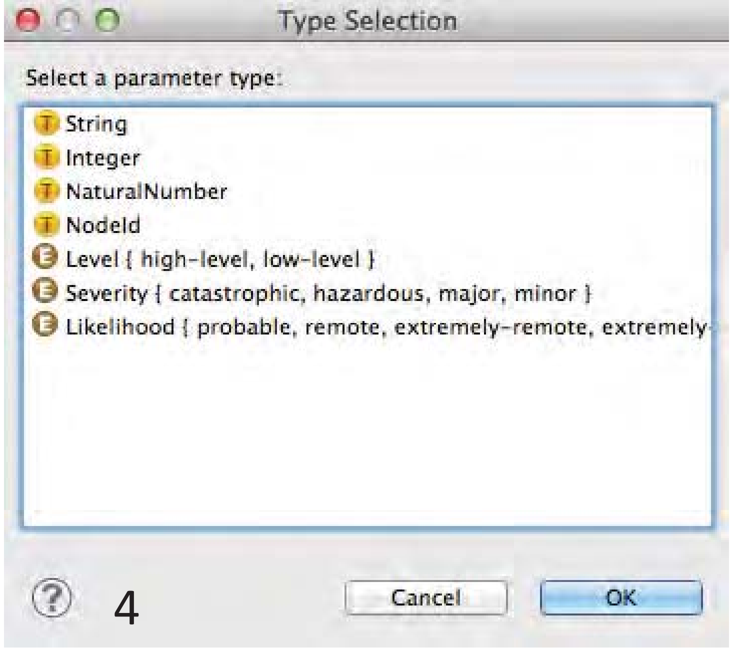
Example (cont.): Editing Metadata—Creating Attributes and Parameters

3. Name the new attribute—in this case, “risk”. Click Add to open the parameter Type Selection box.
4. In the Type Selection box, choose each parameter—Likelihood and Severity—and click Add to add each one to the list of parameters in the Add Attribute box.



Name: risk

Parameters:

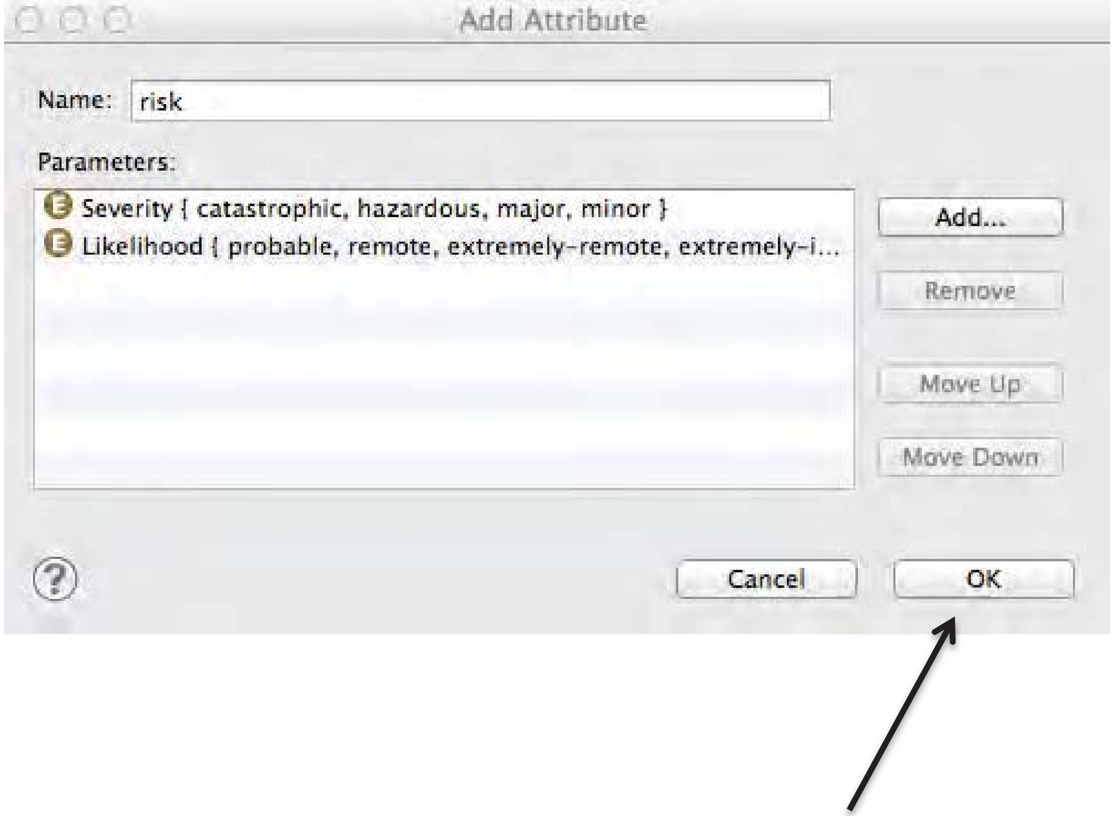


Select a parameter type:

- String
- Integer
- NaturalNumber
- NodeId
- Level { high-level, low-level }
- Severity { catastrophic, hazardous, major, minor }
- Likelihood { probable, remote, extremely-remote, extremely

Example (cont.): Editing Metadata—Creating Attributes and Parameters

5. When all parameters have been added to the Add Attribute list, click OK.



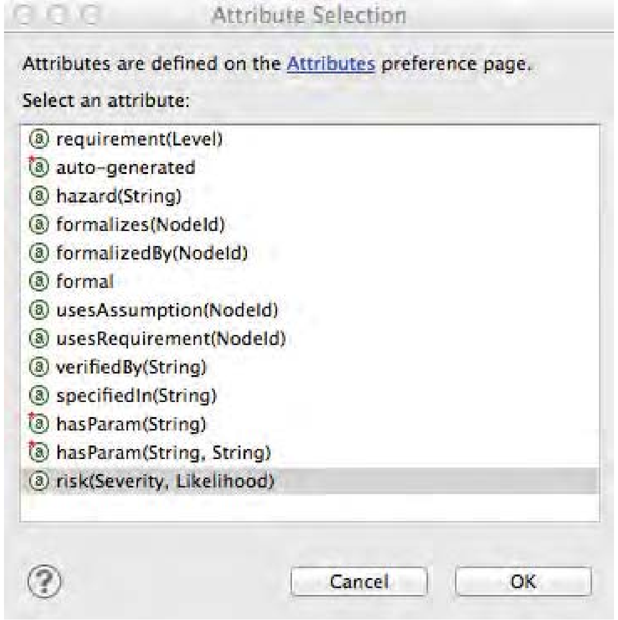
Name: risk

Parameters:

- Severity { catastrophic, hazardous, major, minor }
- Likelihood { probable, remote, extremely-remote, extremely-i...

Example (cont.): Editing Metadata—Creating Attributes and Parameters

6. Risk now appears in the Attribute Selection list. Select Risk and click OK.
7. In the Edit Attributes box, make the appropriate parameter selections. At this stage, enter the parameter characteristics (e.g. “catastrophic” for the Severity parameter) if they do not already exist. Click OK.
8. Click OK in the node’s Edit Attributes box to add the risk attribute to the node’s metadata.

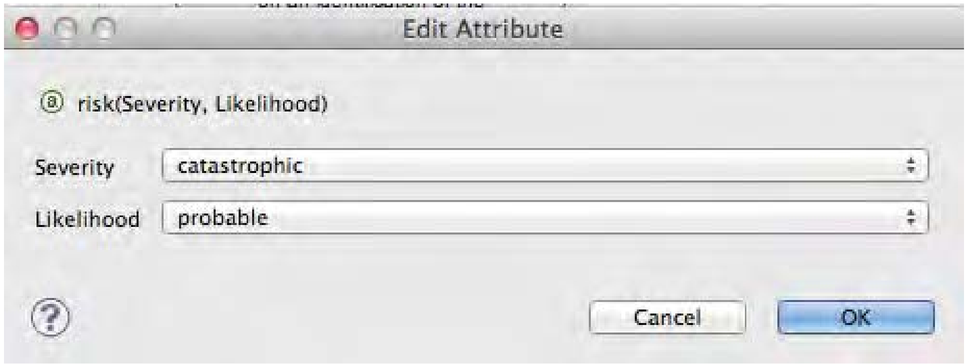


Attributes are defined on the Attributes preference page.

Select an attribute:

- requirement(Level)
- auto-generated
- hazard(String)
- formalizes(NodeId)
- formalizedBy(NodeId)
- formal
- usesAssumption(NodeId)
- usesRequirement(NodeId)
- verifiedBy(String)
- specifiedIn(String)
- hasParam(String)
- hasParam(String, String)
- risk(Severity, Likelihood)

6



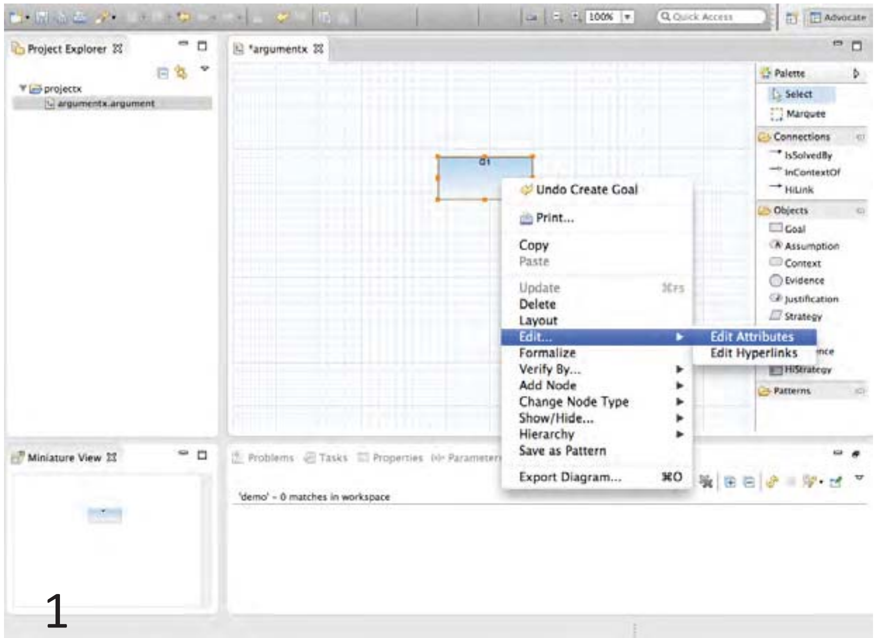
risk(Severity, Likelihood)

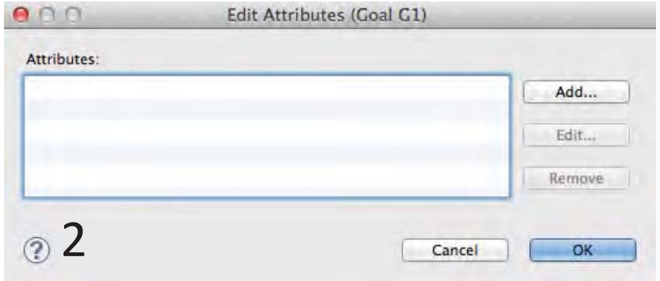
Severity: catastrophic

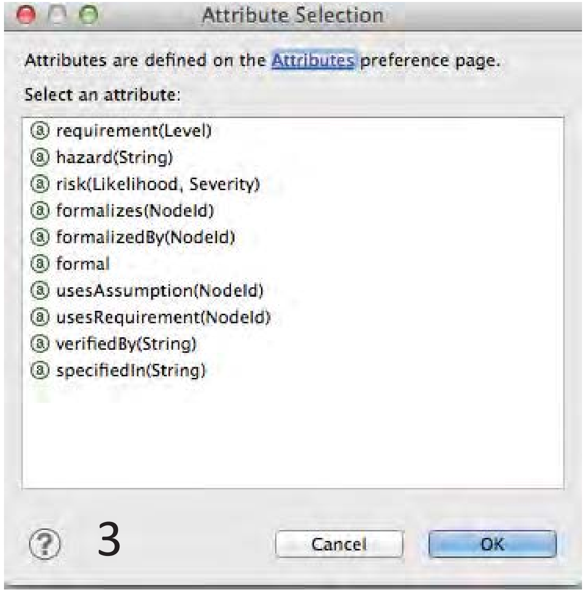
Likelihood: probable

7

Editing Metadata—Adding Existing Attributes







Attributes are defined on the Attributes preference page.

Select an attribute:

- requirement(Level)
- hazard(String)
- risk(Likelihood, Severity)
- formalizes(NodeId)
- formalizedBy(NodeId)
- formal
- usesAssumption(NodeId)
- usesRequirement(NodeId)
- verifiedBy(String)
- specifiedIn(String)

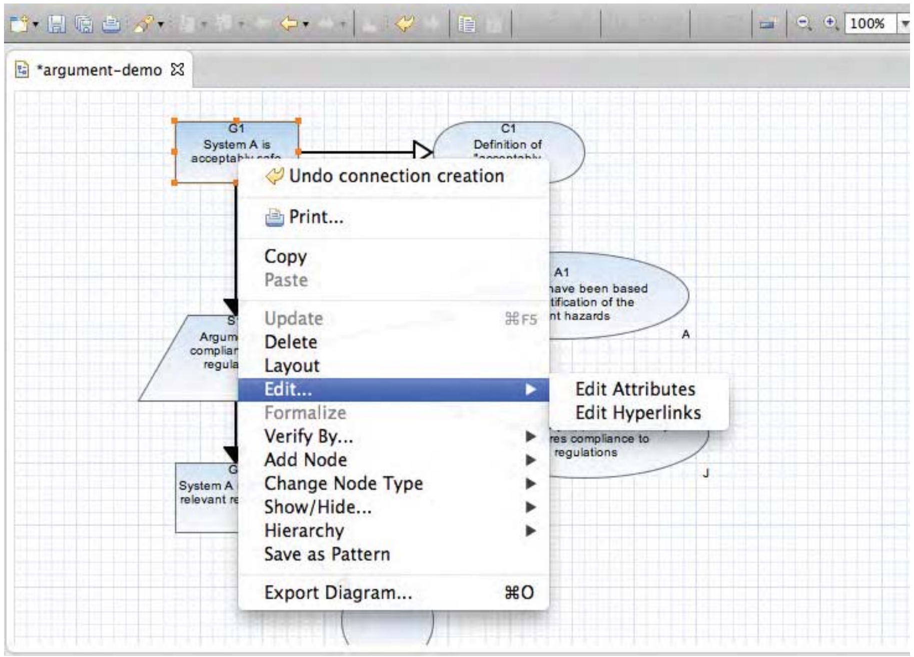
Metadata comprises *attributes,* which in turn are can contain *parameters*.

To add an attribute from the library,

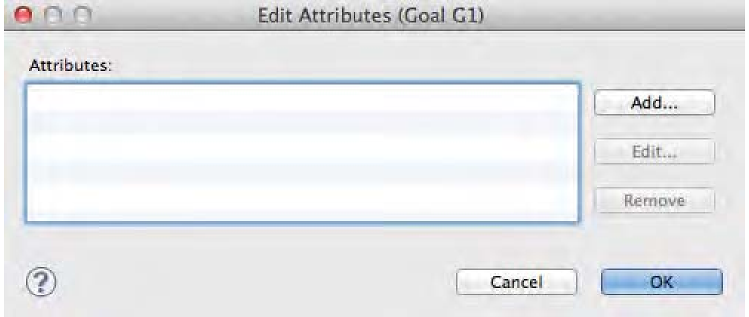
1. Right-click on the node. From the drop-down menu, select Edit > Edit Attributes.
2. In the Edit Attributes box, click Add.
3. In the Attributes Selection box, select the attribute to add and click OK. The added attribute should appear in the node's Edit Attributes box. Click OK.

The attribute will appear in the metadata field of the node's Properties panel.

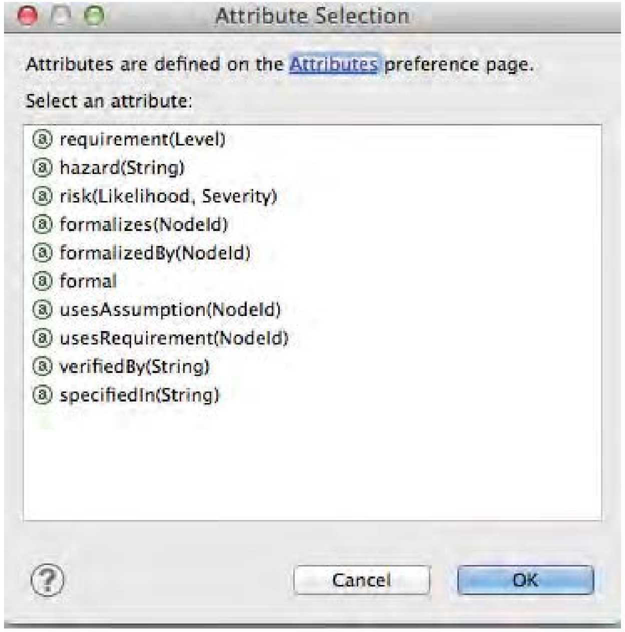
Example: Editing Metadata—Adding Existing Attributes



1



2



Attributes are defined on the Attributes preference page.

Select an attribute:

- requirement(Level)
- hazard(String)
- risk(Likelihood, Severity)
- formalizes(NodeId)
- formalizedBy(NodeId)
- formal
- usesAssumption(NodeId)
- usesRequirement(NodeId)
- verifiedBy(String)
- specifiedIn(String)

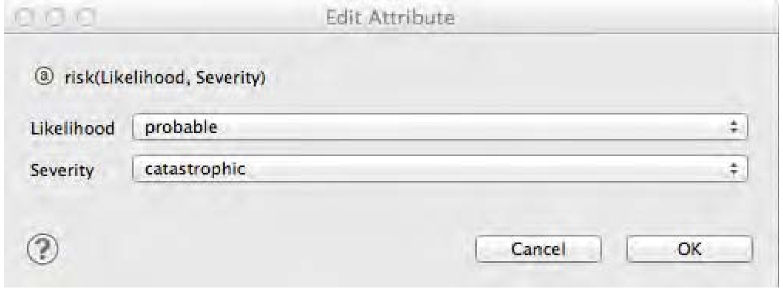
3

For example, to add Risk as an attribute to the G1 node,

1. Select G1. Click Edit and then Edit Attributes.
2. The node's Edit Attributes box should be empty because there are no attributes associated with this node. To add, an attribute, click Add.
3. Select Risk from the list of attributes. The attribute parameters for risk are Likelihood and Severity. Click OK.

Example (cont.): Editing Metadata—Adding Existing Attributes

4

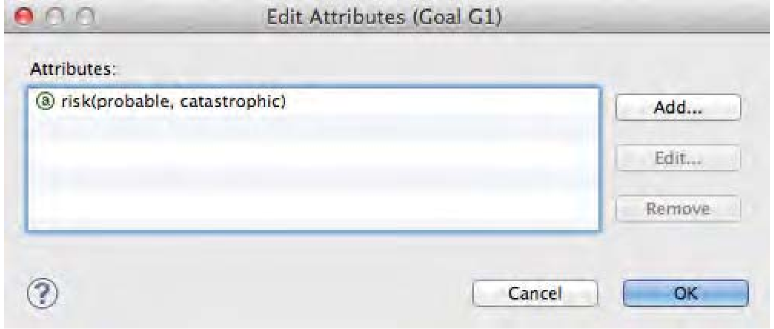


ⓐ risk(Likelihood, Severity)

Likelihood: probable

Severity: catastrophic

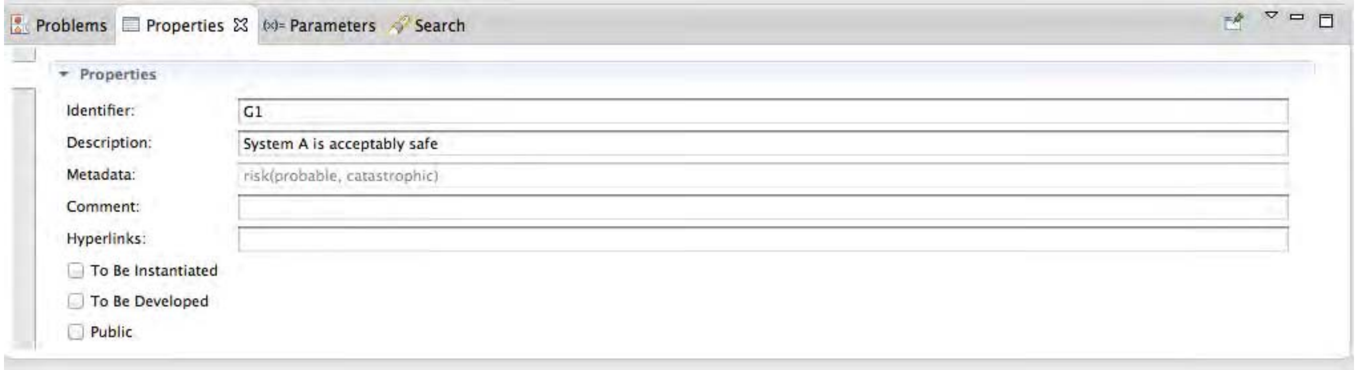
5



Attributes:

ⓐ risk(probable, catastrophic)

6



Identifier: G1

Description: System A is acceptably safe

Metadata: risk(probable, catastrophic)

Comment:

Hyperlinks:

□ To Be Instantiated

□ To Be Developed

□ Public

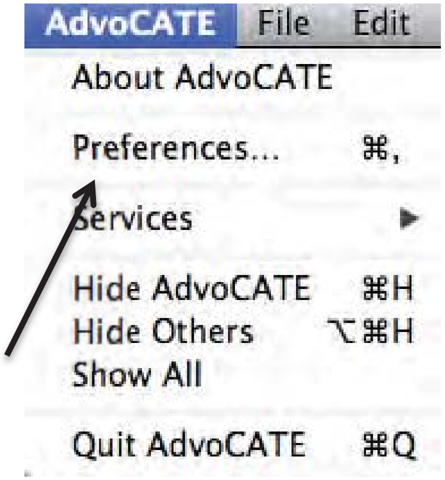
4. In the Edit Attribute box, specify the attribute parameters with the dropdown menus. In this example, the parameters of Likelihood and Severity are characterized as probably and catastrophic, respectively. Click OK.
5. The risk attribute should now appear in the node's Edit Attributes with the specified parameters. Click OK.
6. The risk attribute now appears in the node's metadata field in the Properties panel.

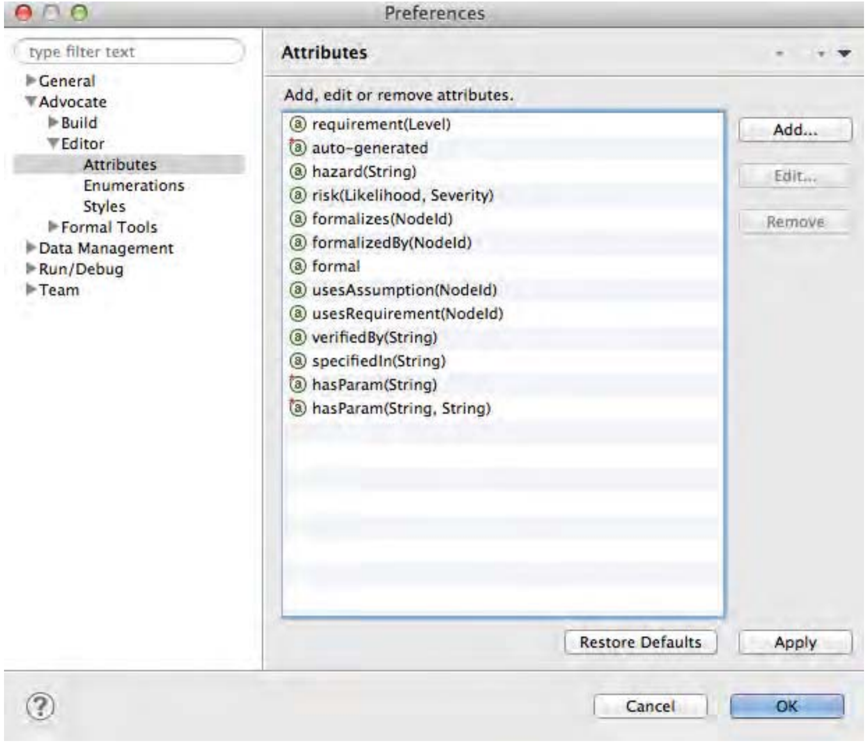
Adding Attributes Directly to the Library

Attributes do not have to be created only when editing a node's metadata.

Add an attribute to the library by:

1. Selecting Preferences from the AdvoCATE tool menu.
2. From the Preferences box, follow the steps as for adding an attribute to a node.
   a. Name the attribute and select its parameters. In the Add Attribute box, click OK.
   b. The new attribute should appear in the Attributes list of the Preferences box. Click OK to close.

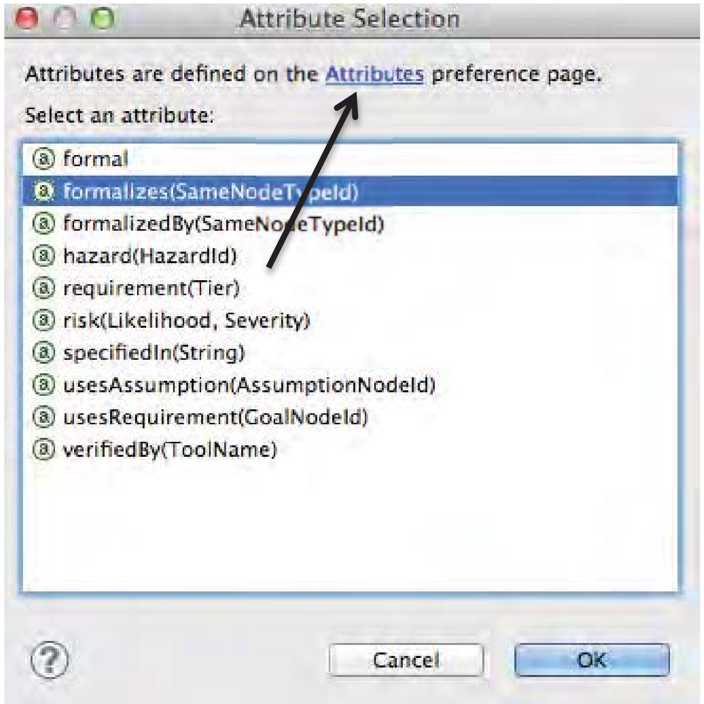




Add, edit or remove attributes.

- requirement(Level)
- auto-generated
- hazard(String)
- risk(Likelihood, Severity)
- formalizes(NodeId)
- formalizedBy(NodeId)
- formal
- usesAssumption(NodeId)
- usesRequirement(NodeId)
- verifiedBy(String)
- specifiedIn(String)
- hasParam(String)
- hasParam(String, String)

Parameter Type ID: SameNodeTypeId



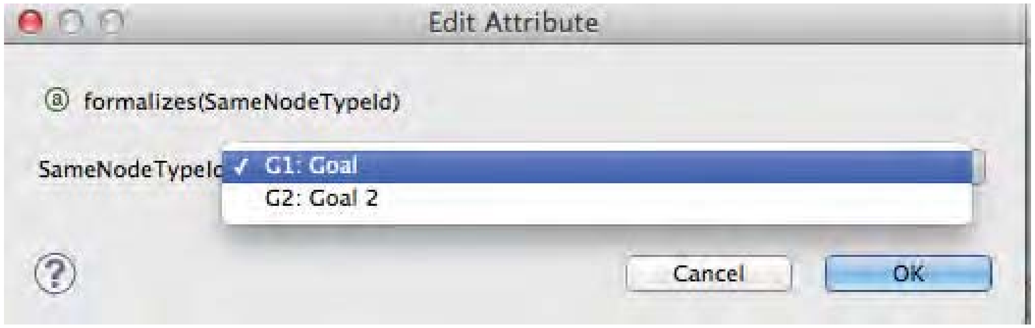
Attributes are defined on the Attributes preference page.

Select an attribute:

- formal
- formalizes(SameNodeTypeId)
- formalizedBy(SameNodeTypeId)
- hazard(HazardId)
- requirement(Tier)
- risk(Likelihood, Severity)
- specifiedIn(String)
- usesAssumption(AssumptionNodeId)
- usesRequirement(GoalNodeId)
- verifiedBy(ToolName)

Each parameter contains an ID that indicates something about which nodes it can be assigned to. Add the SameNodeTypeId attribute to a node to represent relationships between nodes of the same type.

For example, select a goal node, and edit its attributes. Click Add. In the Attribute box, select and add an attribute with (SameNodeTypeID) as its ID. Click OK.

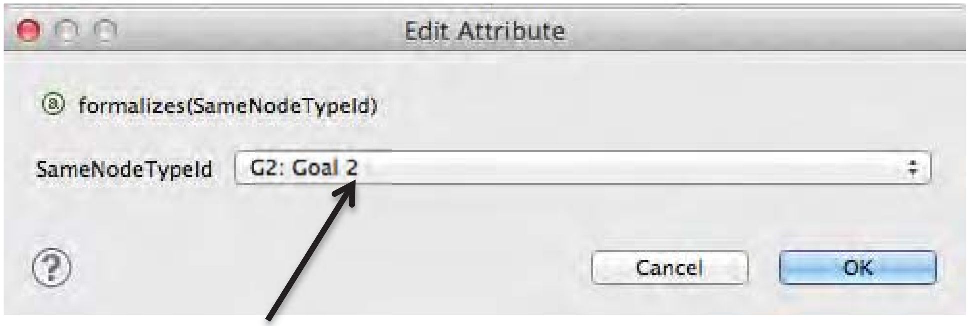


formalizes(SameNodeTypeId)

SameNodeTypeId ✓ G1: Goal
G2: Goal 2

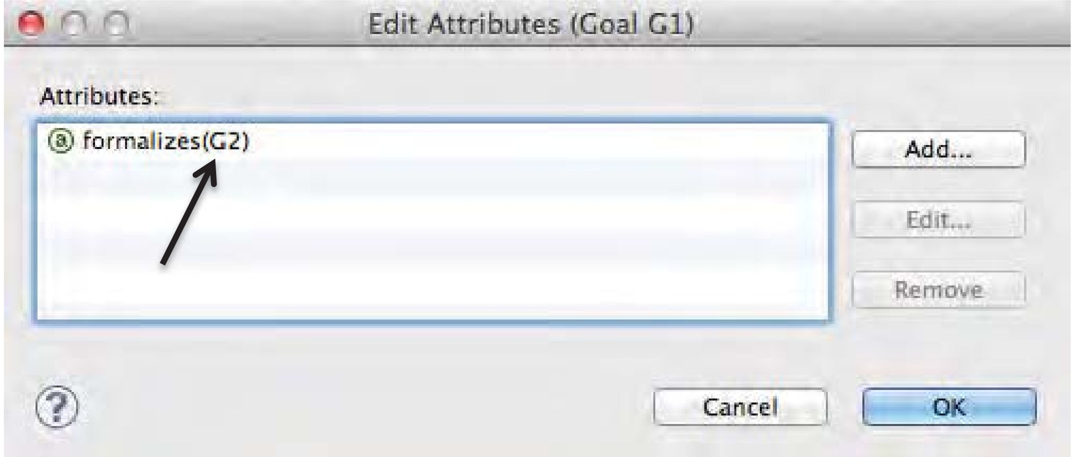
In the Edit Attribute box, notice that this particular attribute ID is specific to goal nodes only—because that is the node type this attribute is assigned to.

Parameter Type ID: SameNodeTypeId (cont.)



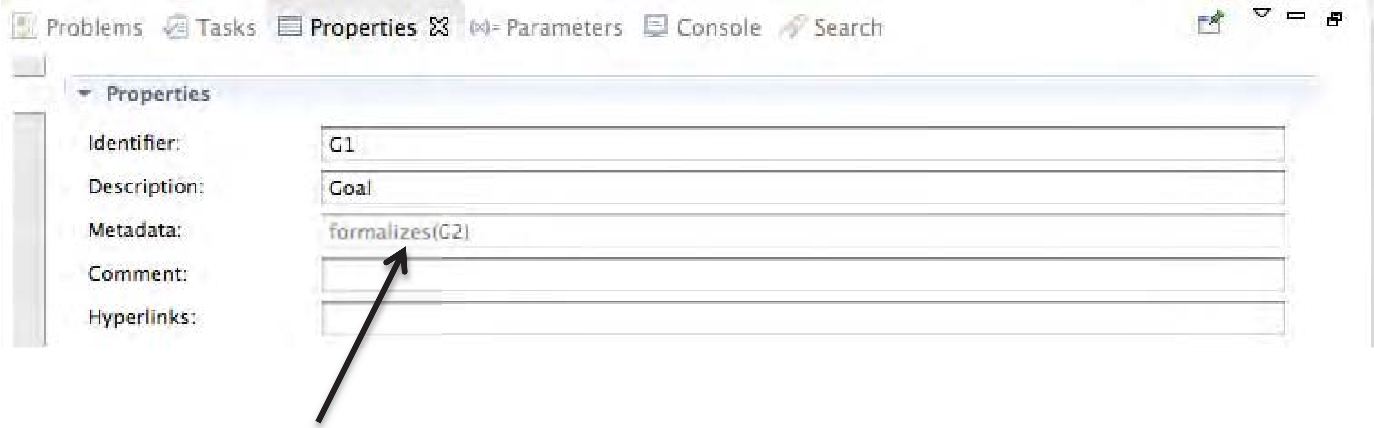
formalizes(SameNodeTypeId)

SameNodeTypeId: G2: Goal 2



Attributes:

formalizes(G2)



Identifier: G1

Description: Goal

Metadata: formalizes(G2)

Comment:

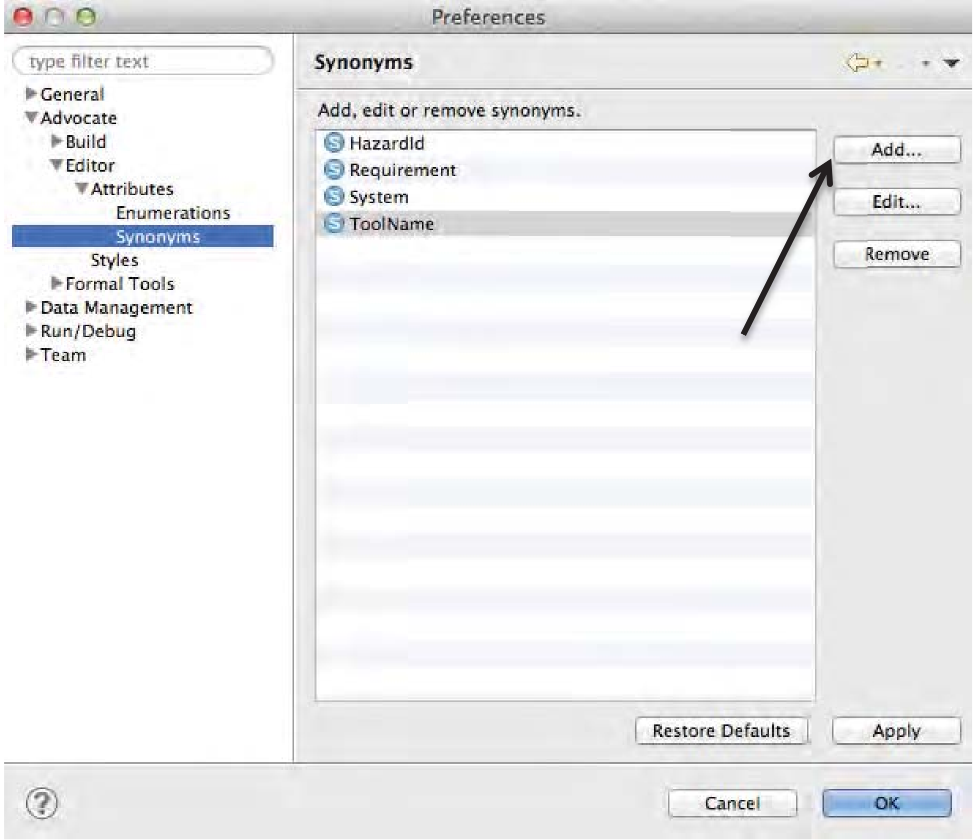
Hyperlinks:

Continuing this example, select a node from the list (in this case, goals) and click OK.

The attribute should appear in the Edit Attributes box. Notice that the attribute ID is the node specified during the previous step (G2). Click OK.

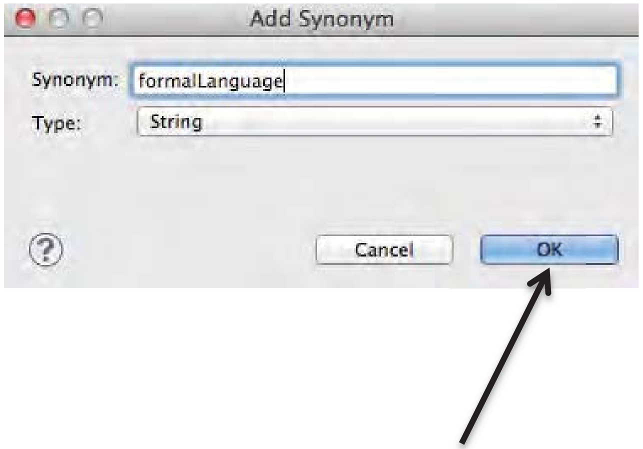
The attribute now appears in the metadata field of the G1 node’s properties.

Attribute Type: Synonyms



Add, edit or remove synonyms.

- HazardId
- Requirement
- System
- ToolName



Synonym: formalLanguage

Type: String

Most of the Attributes covered so far have contained editable enumerations (e.g. likelihood, severity, etc.) but another attribute type is the Synonym.

To add/remove/edit synonyms:

1. Navigate to the Attributes preferences box
2. Under Attributes, select Synonyms
3. Click on an existing Synonym to edit or remove it OR click Add to create a new one. Enter the name of the new Synonym and click OK. It will appearance in the list of Synonyms in the Preferences box.

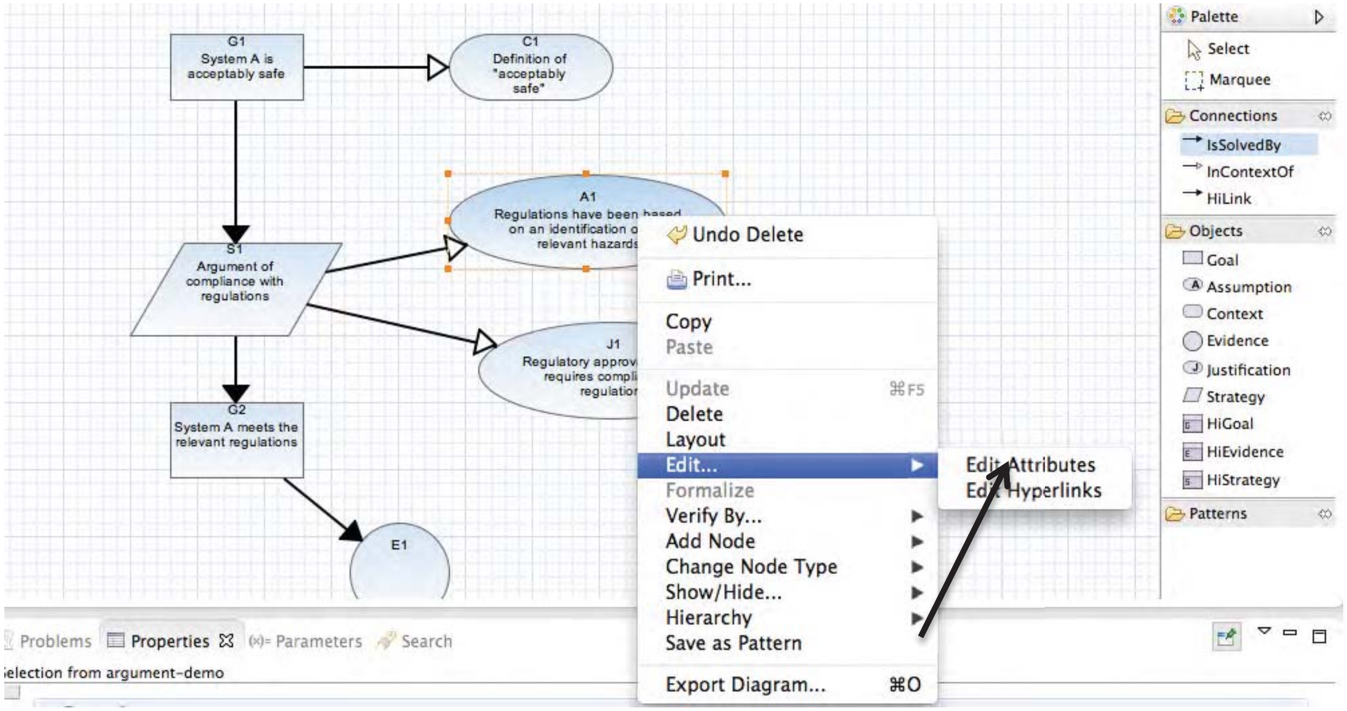
For example, to add a Synonym called formalLanguage, click Add in the Preferences box, type in formalLanguage and specify the Synonym type (here, it is “string”) and click OK.

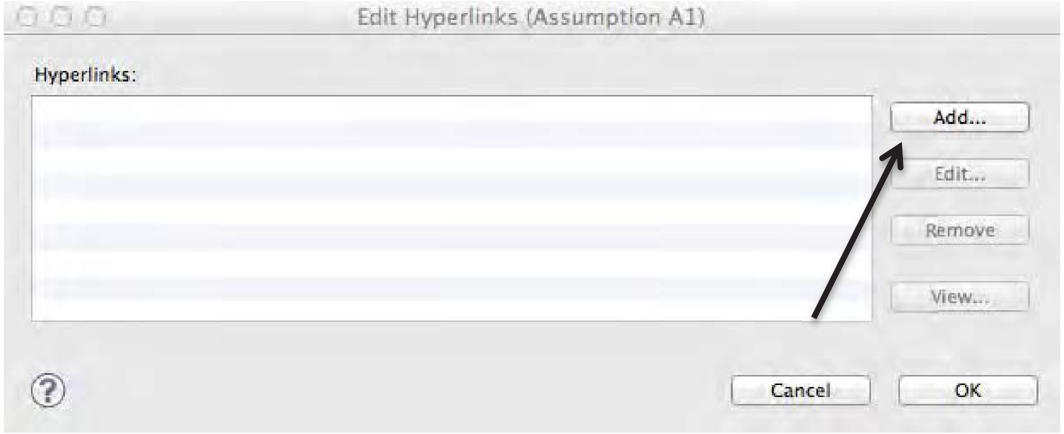
Editing Properties: Adding Hyperlinks to Nodes

Adding a hyperlink to a node is another way to edit its properties.

The method of adding hyperlinks directly to a node is similar to adding attributes.

1. Right-click to select the node. From the menu, select Edit and then Edit Hyperlinks.
2. In the node's Edit Hyperlinks box, add a hyperlink by clicking Add.

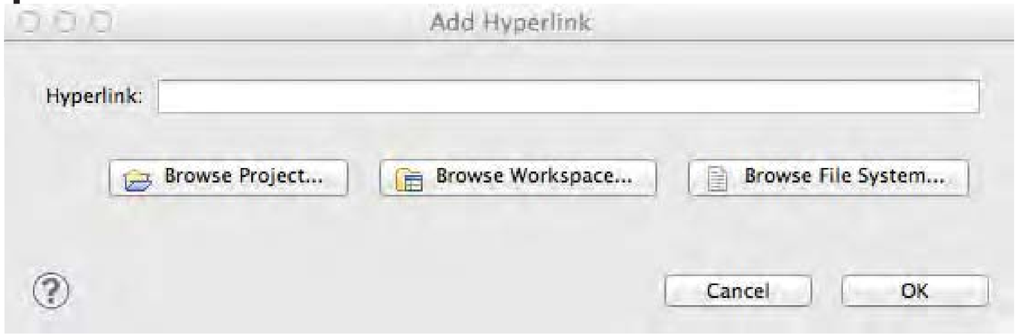


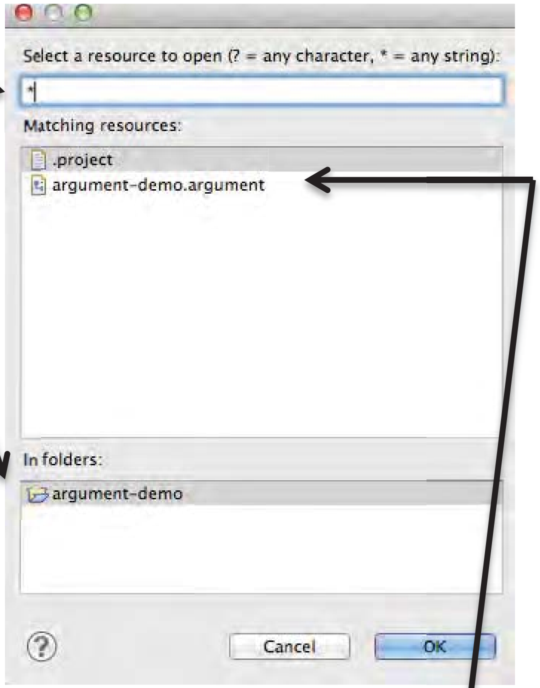


Editing Properties: Adding Hyperlinks to Nodes

3. Add a hyperlink from multiple sources.
   a. Click Browse Projects to link to an existing project resource or folder. In the search box, specify the search terms in either the Resources field or by clicking among the folders. Click OK.
   b. Click Browse Workspace to link to an existing workspace resource or folder. In the search box, specify the search terms in either the Resources field or by clicking among the folders. Click OK.
   c. Click Browse File System to link to local files. Select a file and click OK.
   d. Type in a Web URL directly into the hyperlink field.

Click OK to close the Add Hyperlink box.




Select a resource to open (? = any character, * = any string):
Matching resources:
In folders:
?
Cancel
OK

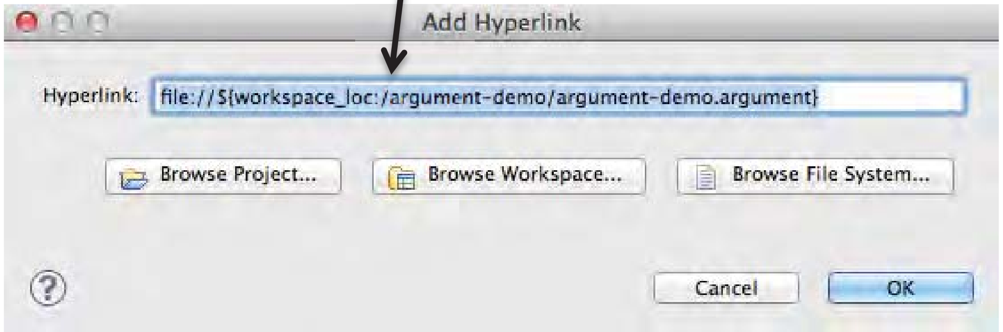
*

.project

argument-demo.argument

argument-demo

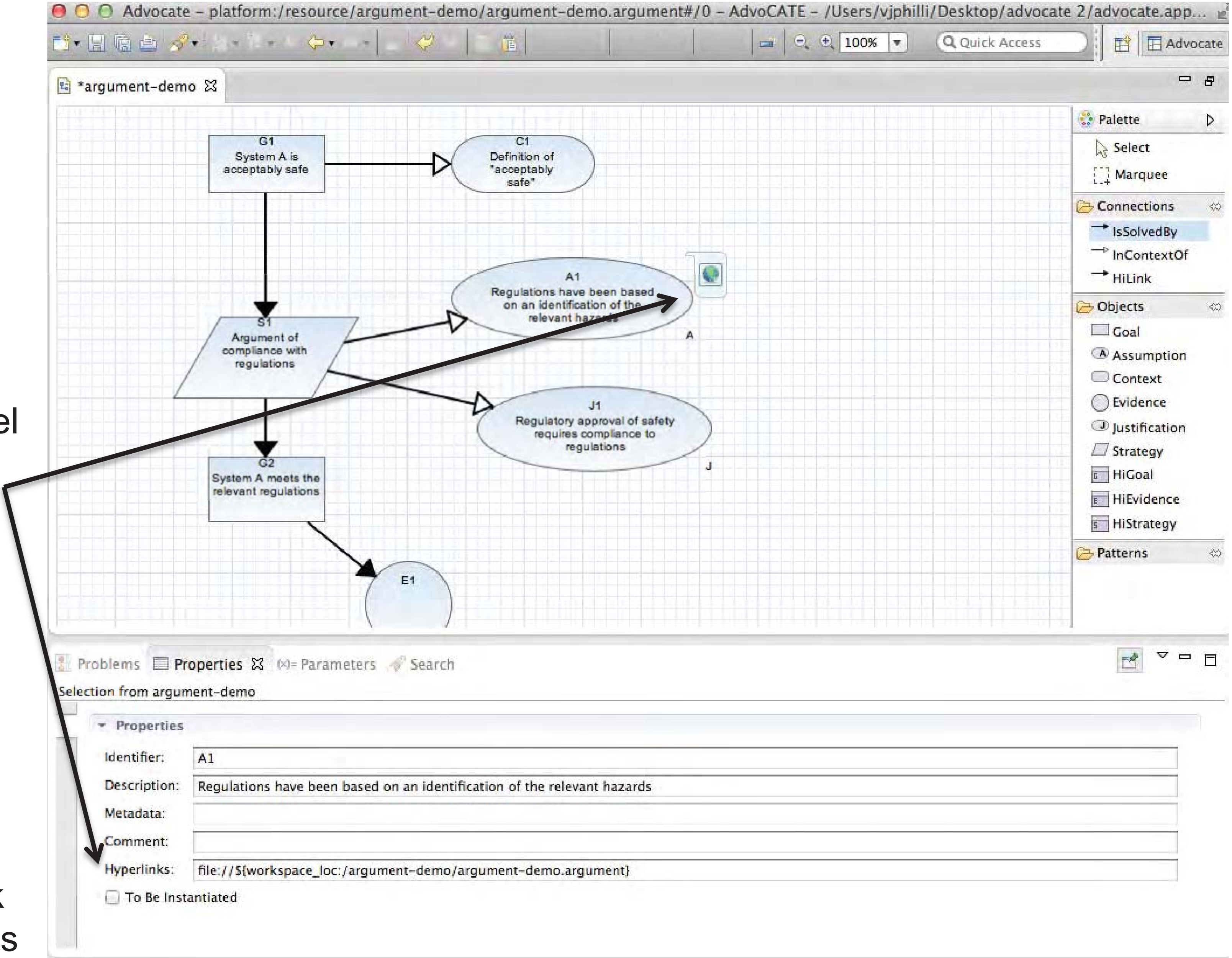
In this example, the user clicked Browse Project. In the search box, the user typed * to show any string of resources and folders associated with that project. The user selected a resource and clicked OK to close the search box. The resource hyperlink appeared in the Hyperlink link field of the Add Hyperlink Box.



Hyperlink: file://${workspace_loc:/argument-demo/argument-demo.argument}

Editing Properties: Adding Hyperlinks to Nodes

4. Click OK to close the Edit Hyperlinks box. The hyperlink should now appear in two places: in the Hyperlinks field in the node's Properties panel and as a globe icon when hovering over the node. Clicking the globe icon will open up a dialogue box to select a link to navigate to. Some hyperlink files will open as a separate tab.


Advocate – platform:/resource/argument-demo/argument-demo.argument#/0 – AdvoCATE – /Users/vjphilli/Desktop/advocate 2/advocate.app...
100%
Quick Access
Advocate
*argument-demo
G1
System A is acceptably safe
C1
Definition of "acceptably safe"
A1
Regulations have been based on an identification of the relevant hazards
A
S1
Argument of compliance with regulations
J1
Regulatory approval of safety requires compliance to regulations
J
G2
System A meets the relevant regulations
E1
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
HiGoal
HiEvidence
HiStrategy
Patterns
Problems
Properties
(x)= Parameters
Search

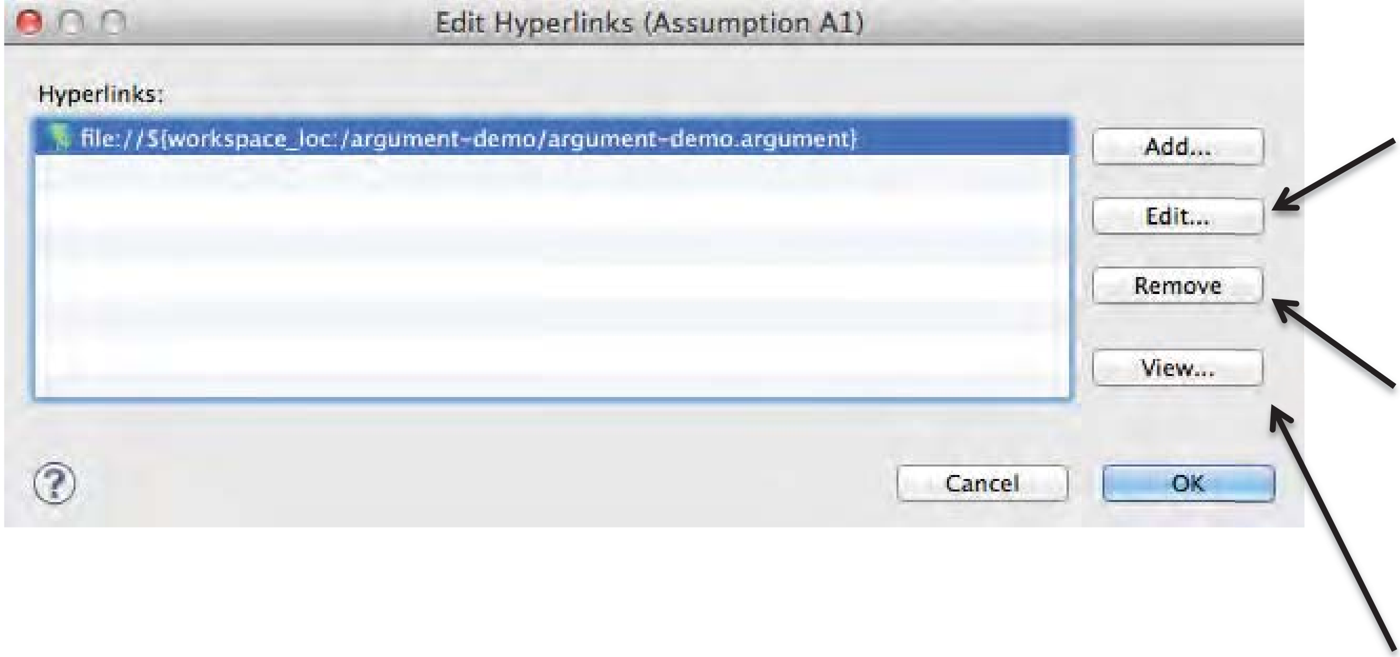
Selection from argument-demo

Properties

Identifier:	A1
Description:	Regulations have been based on an identification of the relevant hazards
Metadata:	
Comment:	
Hyperlinks:	file://${workspace_loc:/argument-demo/argument-demo.argument}

□ To Be Instantiated

Editing Properties: Editing/Viewing/Removing Hyperlinks to Nodes



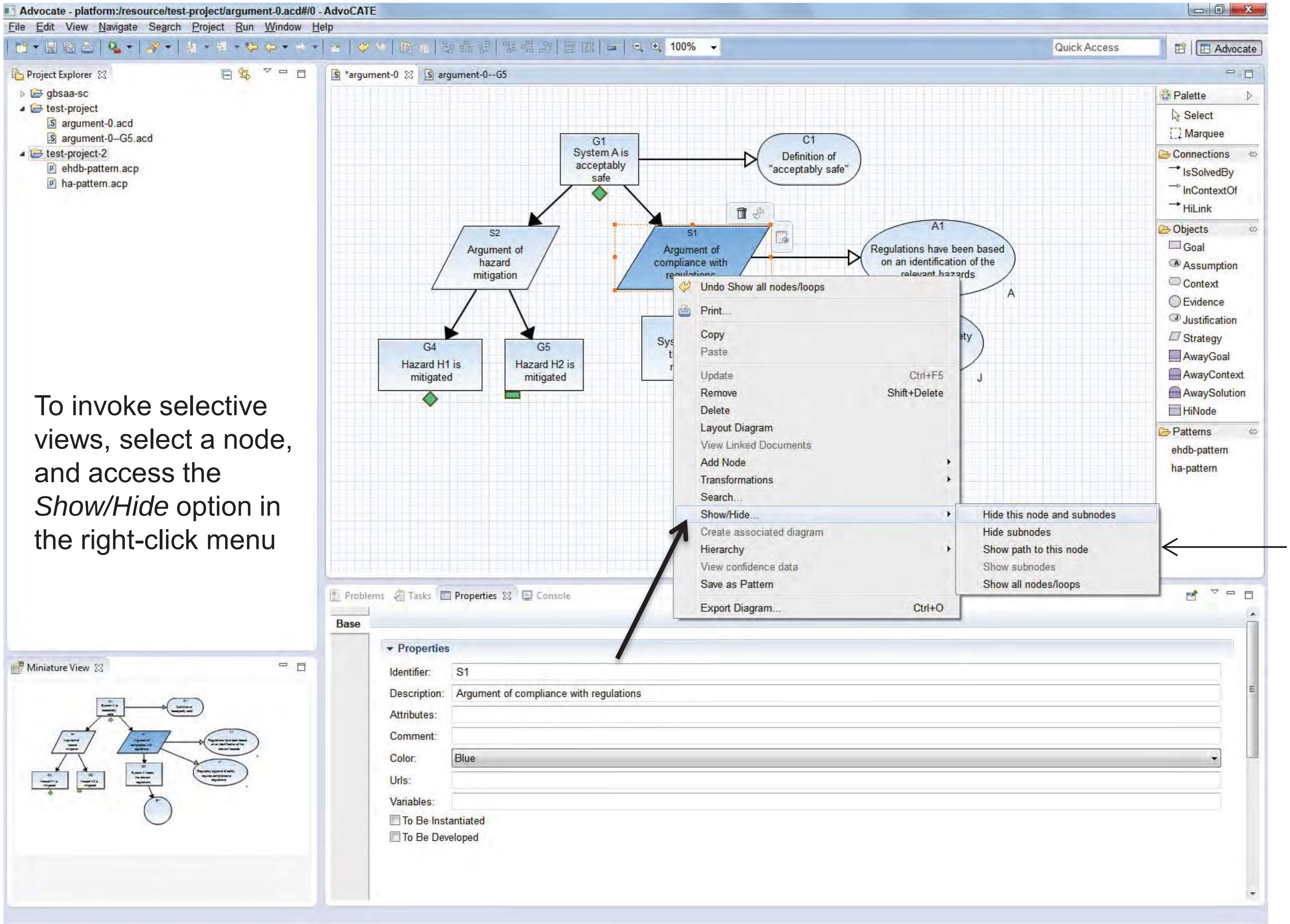
file://${workspace_loc:/argument-demo/argument-demo.argument}

Once a hyperlink has been added to a node, it can be edited, removed, or viewed in a separate project tab.

- To edit a hyperlink, select the hyperlink within the node’s Edit Hyperlinks box, and click Edit.
- To delete a hyperlink, select the hyperlink within the node’s Edit Hyperlinks box, and click Remove.
- To view the hyperlink in the separate project tab, select the hyperlink within the node’s Edit Hyperlinks box and click View.

Selective Views

Selective views provide queries of hiding / showing specific views so as to manage argument structure complexity



Identifier: S1

Description: Argument of compliance with regulations

Attributes:

Comment:

Color: Blue

Urls:

Variables:

☐ To Be Instantiated

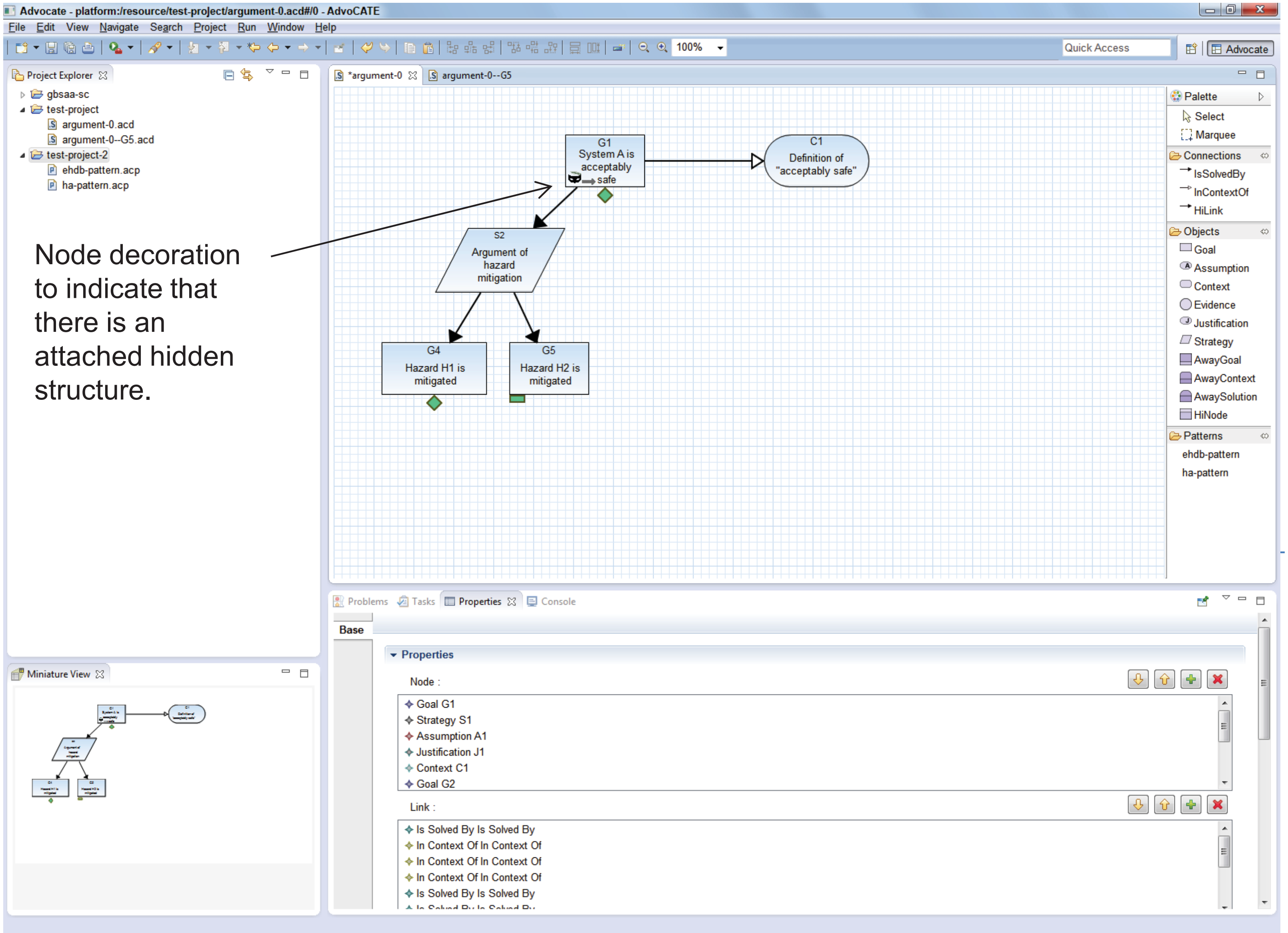
☐ To Be Developed

To invoke selective views, select a node, and access the *Show/Hide* option in the right-click menu

Five views are currently available

Selective Views

On selecting a view, a node or its parent, as appropriate, is decorated to indicate that the view includes hidden structures



Properties

Node :

- Goal G1
- Strategy S1
- Assumption A1
- Justification J1
- Context C1
- Goal G2

Link :

- Is Solved By Is Solved By
- In Context Of In Context Of
- In Context Of In Context Of
- In Context Of In Context Of
- Is Solved By Is Solved By

Node decoration to indicate that there is an attached hidden structure.

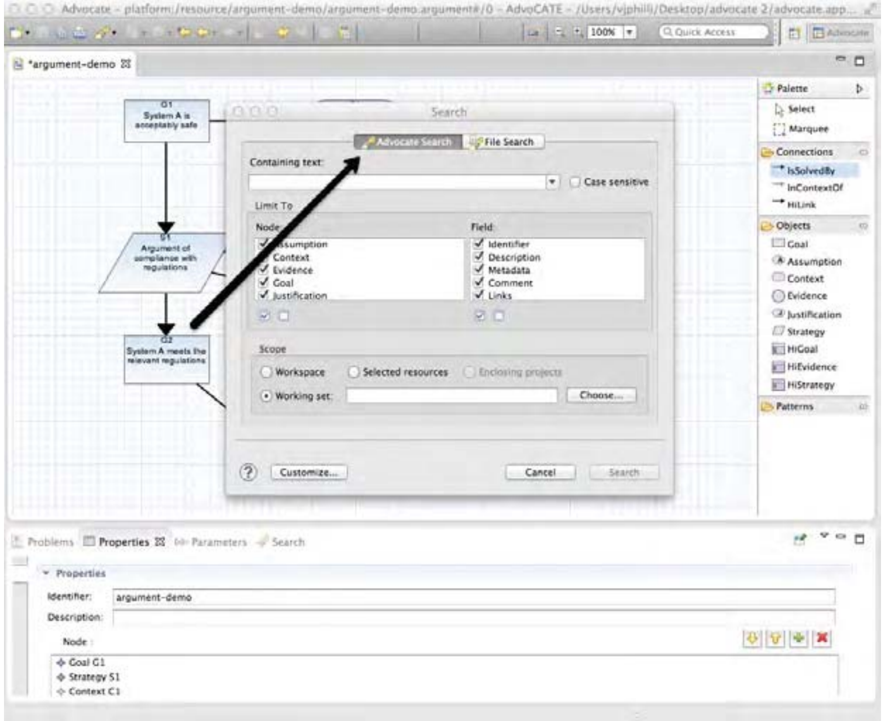
To show a hidden structure:

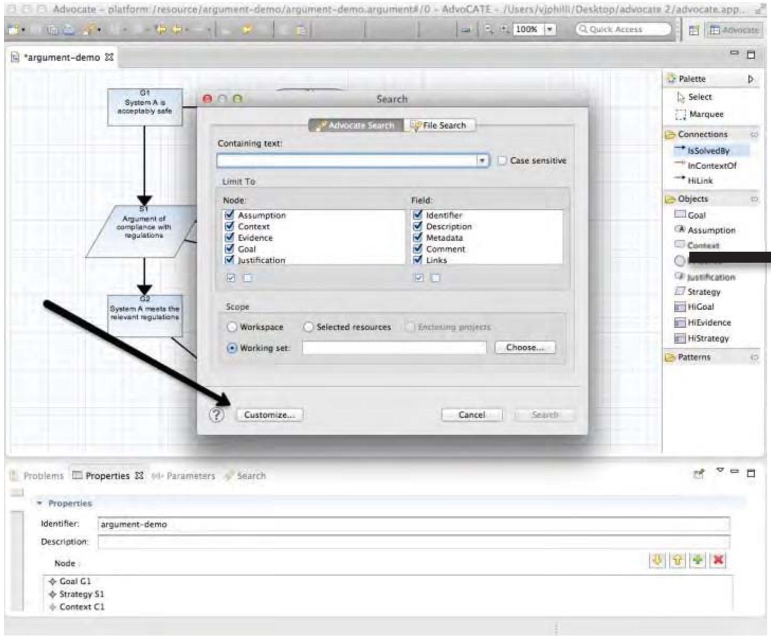
1. select the decorated node,
2. access the *show/hide* option from the right-click menu
3. select the appropriate view

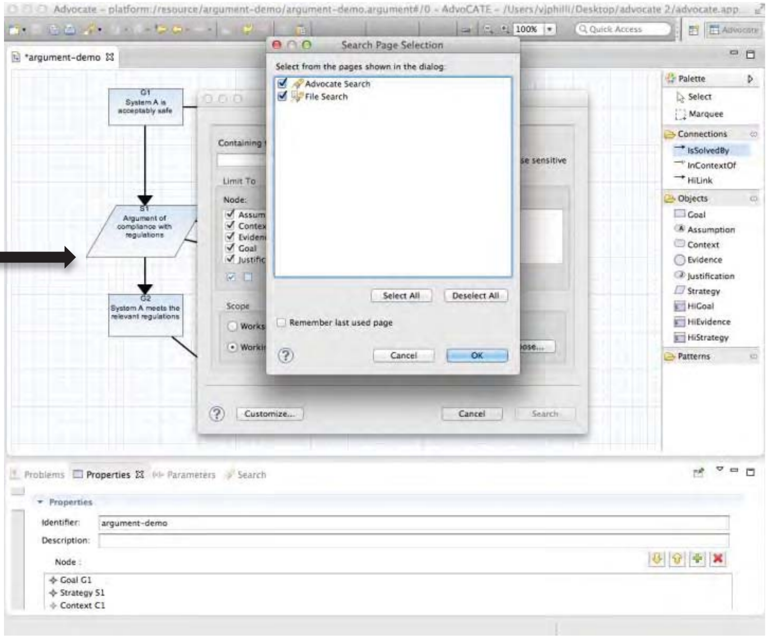
Search

To search for keywords appearing in the project's text, open the Search dialogue box by selecting the Search tab in the application toolbar (or using the keyboard shortcut). Within the Search dialogue box, choose to search within the AdvoCATE application or within local files by

1. Clicking the Customize button at the bottom left of the dialogue box and choosing an Advocate Search or local File Search OR
2. Selecting either tab at the top of the box.





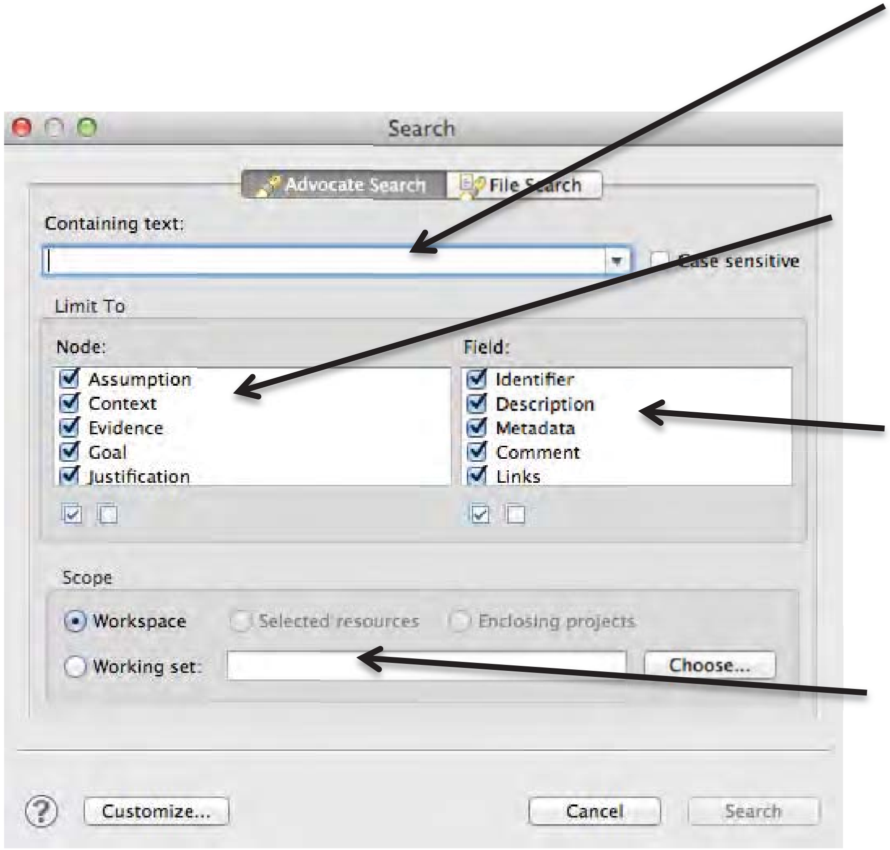


Search in AdvoCATE

Limit the scope of the AdvoCATE search by:

- Limiting the search to specific node types by checking the appropriate boxes under Node in the search dialogue box.
- Limiting the search to specific fields within the node parameters by checking the appropriate boxes under Field in the search dialogue box.
- Defining the scope of the search to be within the
  - The Workspace
  - Selected Resources
  - Enclosing Projects
  - Defined Working Sets

One the search terms have been entered, click Search. Search results will appear in the Search panel below the canvas.



Containing text: □ Case sensitive

Limit To

Node:

☑ Assumption
☑ Context
☑ Evidence
☑ Goal
☑ Justification

Field:

☑ Identifier
☑ Description
☑ Metadata
☑ Comment
☑ Links

Scope

● Workspace ○ Selected resources ○ Enclosing projects

○ Working set:

Enter Search terms here

Narrow the Search to specific node types by checking the boxes besides each Node

Narrow the Search to specific node parameters by checking the boxes besides each Field

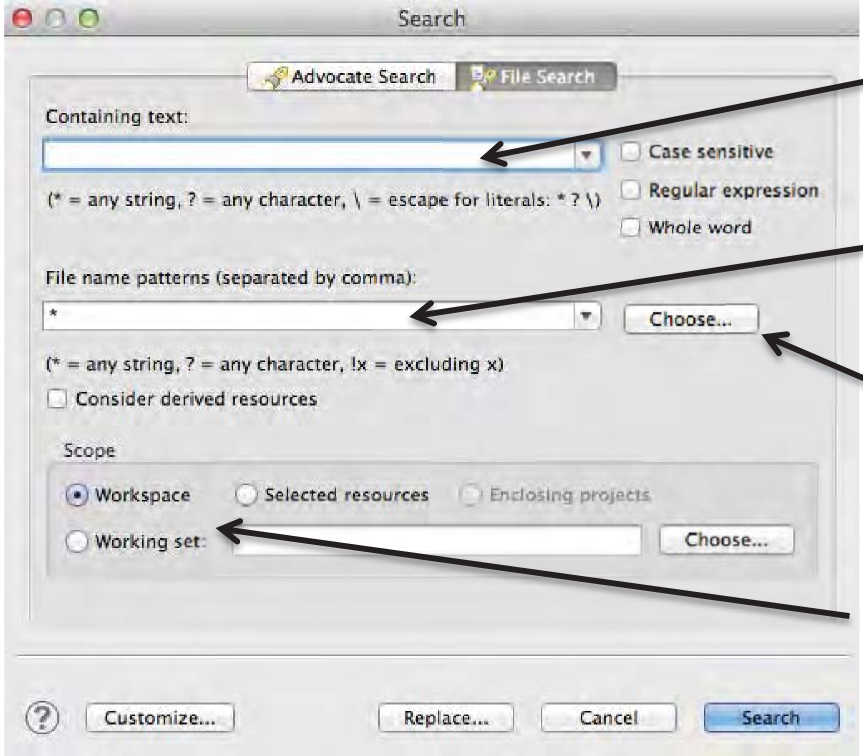
Define the Scope of the search

Search in Files

Limit the scope of the File search by:

- Limiting the search to specific File name patterns. Separate terms with commas.
- Limiting the search to specific file extensions by clicking Choose and selecting the appropriate file extension type.
- Defining the scope of the search to be within the
  - The Workspace
  - Selected Resources
  - Enclosing Projects
  - Defined Working Sets

One the search terms have been entered, click Search. Search results will appear in the Search panel below the canvas.



Containing text:

(* = any string, ? = any character, \ = escape for literals: * ? \)

☐ Case sensitive

☐ Regular expression

☐ Whole word

File name patterns (separated by comma): *

(* = any string, ? = any character, !x = excluding x)

☐ Consider derived resources

Scope

● Workspace ○ Selected resources ○ Enclosing projects

○ Working set:

Enter Search terms here

Enter terms here to search through File name patterns

Choose which file extensions to search through

Define the Scope of the search

Structuring Arguments

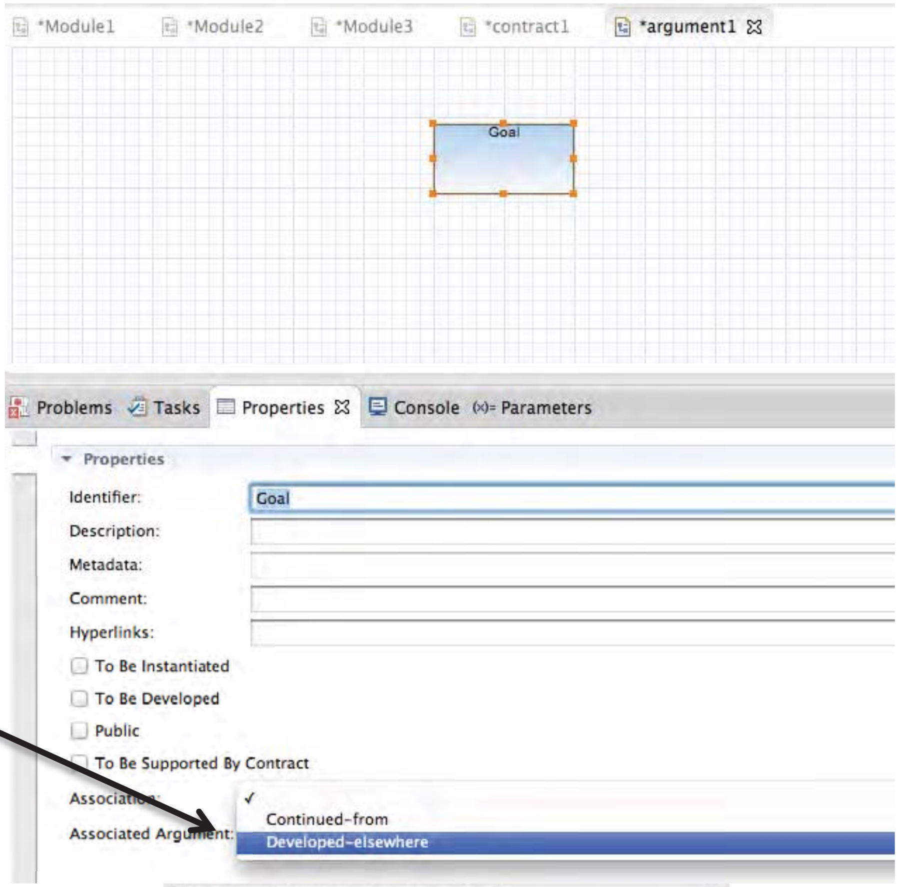
Associated Diagrams

- Associated diagrams manage argument structure complexity by splitting a structure across multiple canvases.
- An association connects source nodes to target nodes through an association property within the nodes' Properties.
  - The target node will show a Continued-from association.
  - The source node will show a Developed-elsewhere association.

Example: Associated Diagrams between Existing and New Arguments

To create an association between an existing source node and new target node,

1. Select the source node and, in its Properties, click the Association drop-down menu and select “Developed-elsewhere”. A green box will appear in the lower left of the node.
2. An Associated Argument Selection box will appear with a field to search through existing arguments with that node name and type.If none exist, click <New> and then OK.



Properties

Identifier: Goal

Description:

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

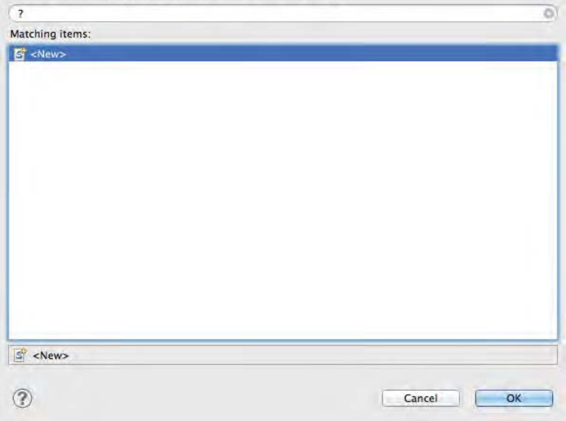
□ To Be Developed

□ Public

□ To Be Supported By Contract

Association: ✓

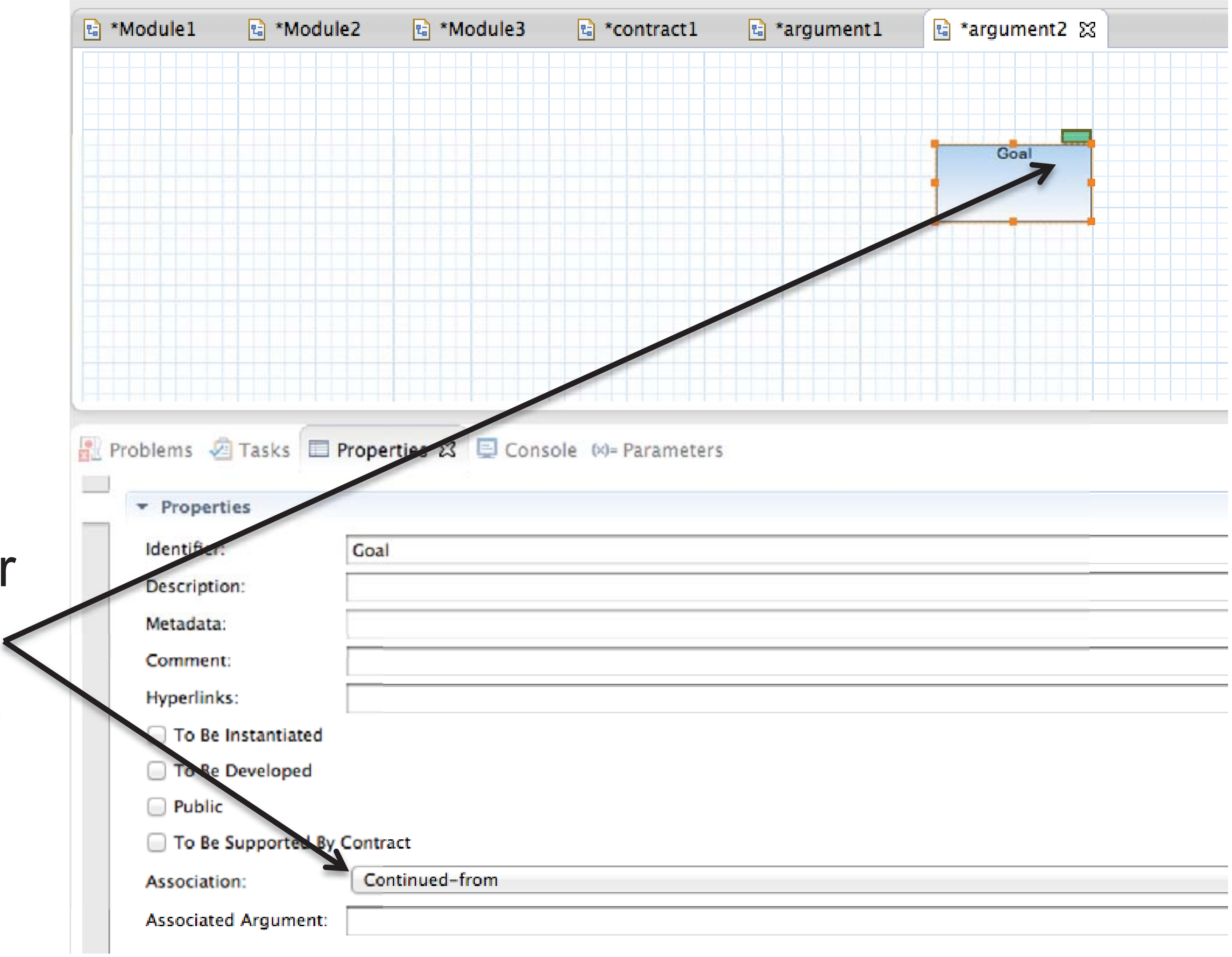
Associated Argument:



Example (cont.): Associated Diagrams between Existing and New Arguments

3. Select the project in which to create the target node and name the target argument. Click Finish.

- The new node should now show a green box on the upper right, indicating that it is associated with another node. The node's Properties indicate that it is "Continued-from" another node.



Properties

Identifier: Goal

Description:

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

□ To Be Developed

□ Public

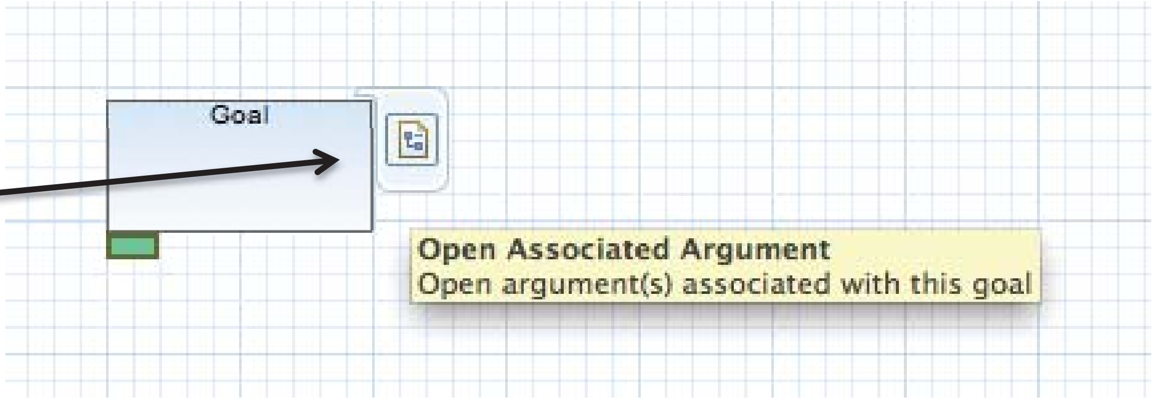
□ To Be Supported By Contract

Association: Continued-from

Associated Argument:

Example (cont.): Associated Diagrams between Existing and New Arguments

- Clicking the icon on the upper right of either node will direct to to its associated counterpoint.

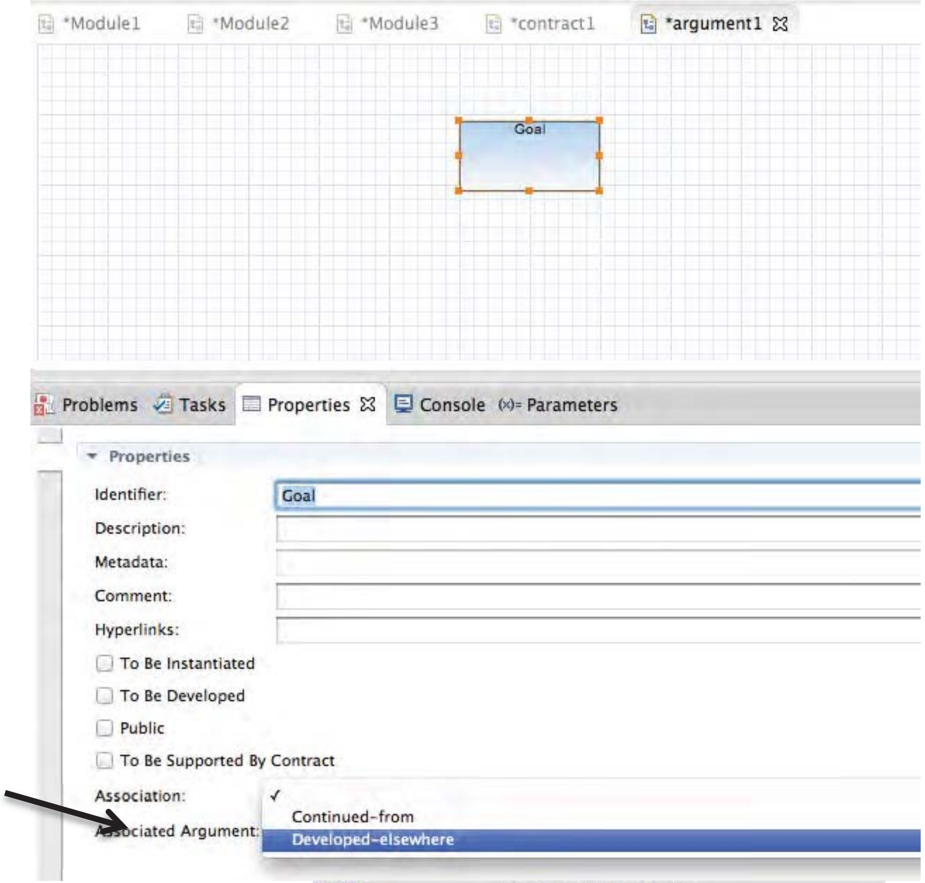


Open Associated Argument
Open argument(s) associated with this goal

Example: Associated Diagrams between Two Existing Arguments

To create an association between an existing source node and an existing target node,

1. Create two nodes of the same name, one in each of two arguments. For example, G1 in argument1.argument and G1 in argument2.argument.
2. Select the source node and, in its Properties, click the Association drop-down menu and select "Developed-elsewhere". A green box will appear in the lower left of the node.
3. An Associated Argument Selection box will appear with a field to search through existing arguments with that node name and type. Select the appropriate argument (in this example, argument2.argument and click OK.



Properties

Identifier: Goal

Description:

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

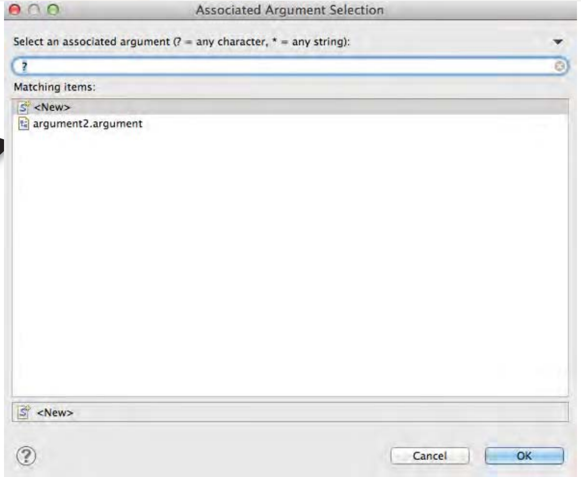
□ To Be Developed

□ Public

□ To Be Supported By Contract

Association: ✓

Associated Argument:



Select an associated argument (? = any character, * = any string):

?

Matching items:

<New>

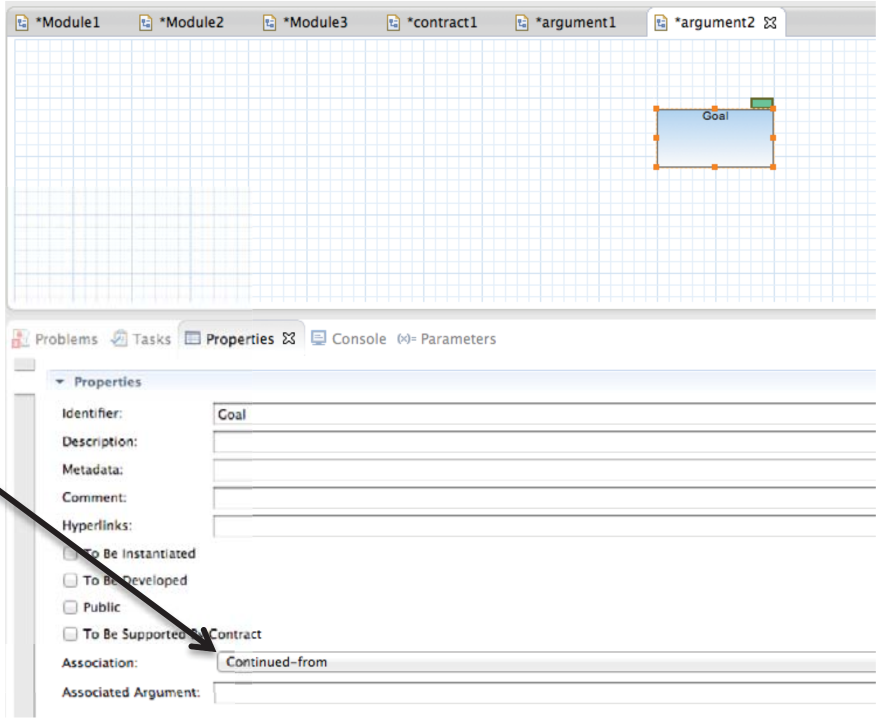
argument2.argument

<New>

Example: Associated Diagrams between Two Existing Arguments (cont.)

4. AdvoCATE will automatically open the target argument’s canvas. In the Association drop-down menu of the target node’s Properties, select “Continued-from”. The node should now show a green box on the upper right, indicating that it is associated with another node.

- Clicking the icon on the upper right of either node will direct to to its associated counterpoint.



Properties

Identifier: Goal

Description:

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

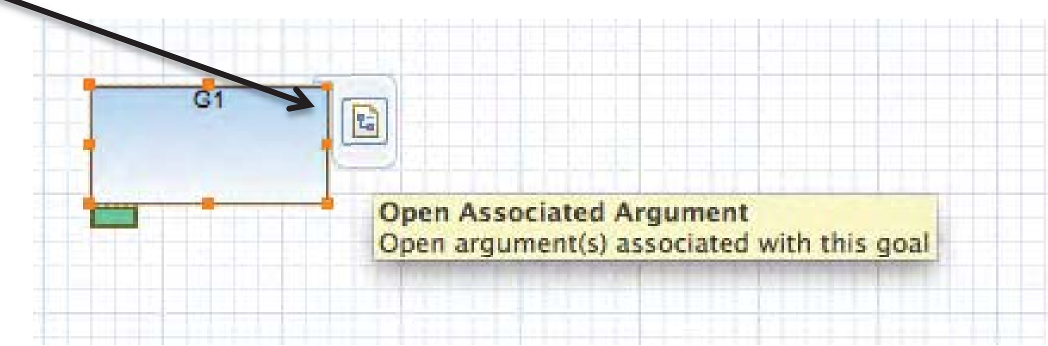
□ To Be Developed

□ Public

□ To Be Supported by Contract

Association: Continued-from

Associated Argument:



Open Associated Argument
Open argument(s) associated with this goal

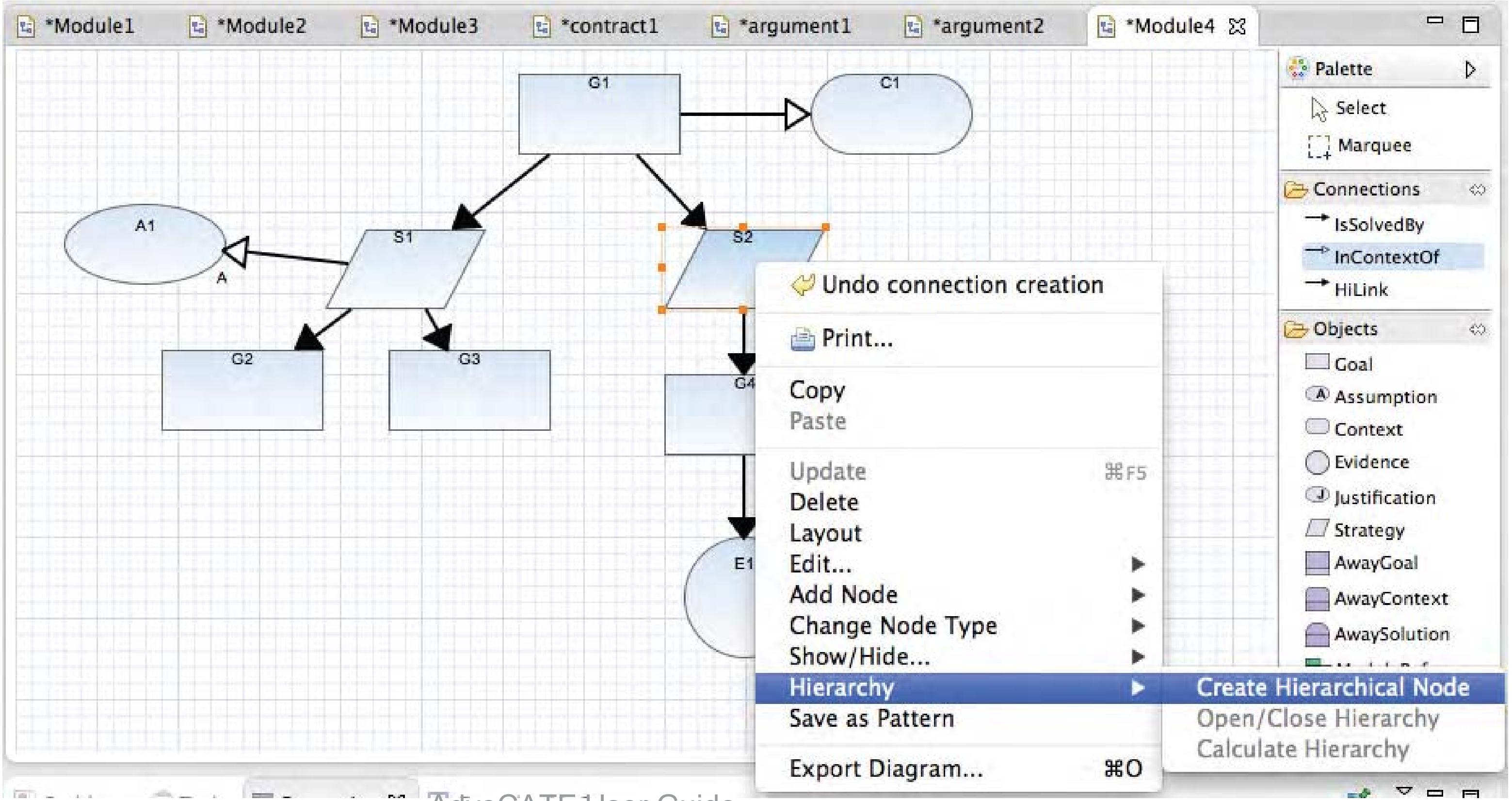
Hierarchy

Another method to reduce a diagram's complexity is to employ Hierarchy to create a recursive group of meaningful subsets of the argument structure.

To add hierarchy:

1. Select the root node. Optionally, select additional nodes to include in the hierarchy structure. Right-click on the root node.
2. Select Hierarchy→ Create Hierarchical Node

Hierarchy currently cannot be applied to patterns or modules.



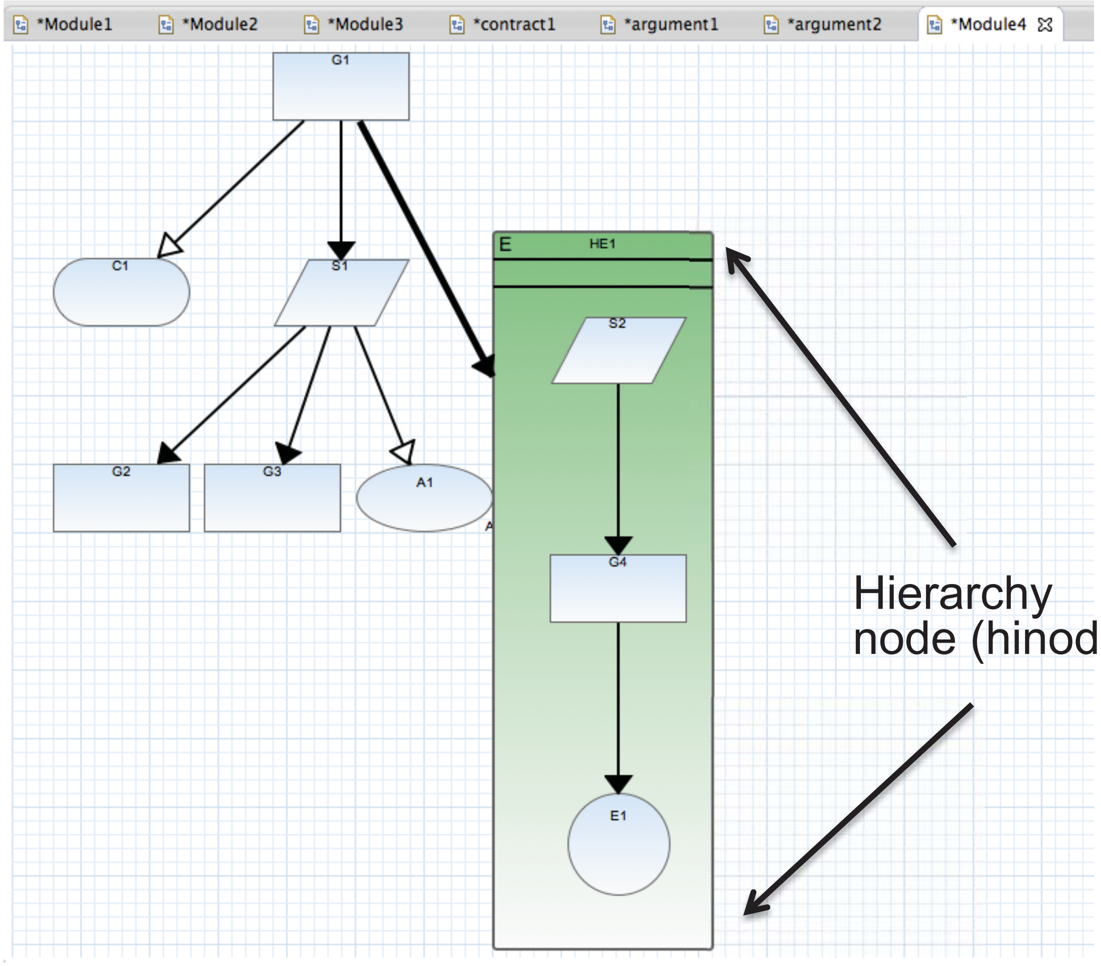
Creating Hierarchy

The structure undergoes auto-layout whenever a hierarchical node (hinode) is created.

AdvoCATE currently computes three kinds of hinodes, subject to certain conditions:

- hierarchical goals (higoals), annotated as **G**
- hierarchical strategies annotated as **S**
- hierarchical evidence annotated as **E**

The hinode type is determined by the nodes contained within the hierarchical structure.



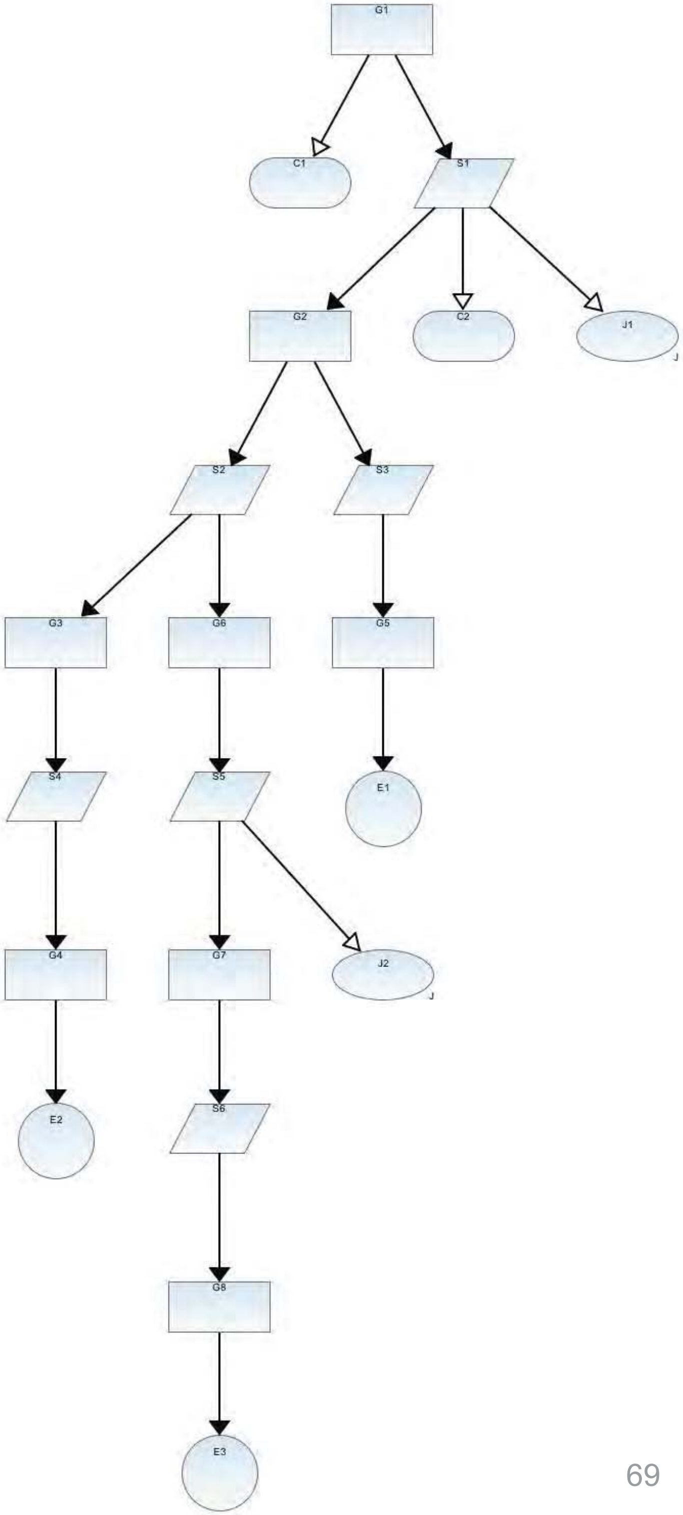
Hierarchy node (hinode)

Example: Creating Hierarchical Structures

The next several slides will walk through creating hierarchical goals, strategies, and evidence structures based on the complex diagram at right.

Note that the rectangles are goal nodes, the oblong shapes are context nodes, the ovals are justification nodes, the circles are evidence nodes, and the parallelograms are strategy nodes.

At each step you select the root of the fragment which will be turned
into a hinode and, optionally, the nodes that come just after thefragment to be included in the hierarchy (the delimiting nodes). Multiple nodes can be selected using Command-click.

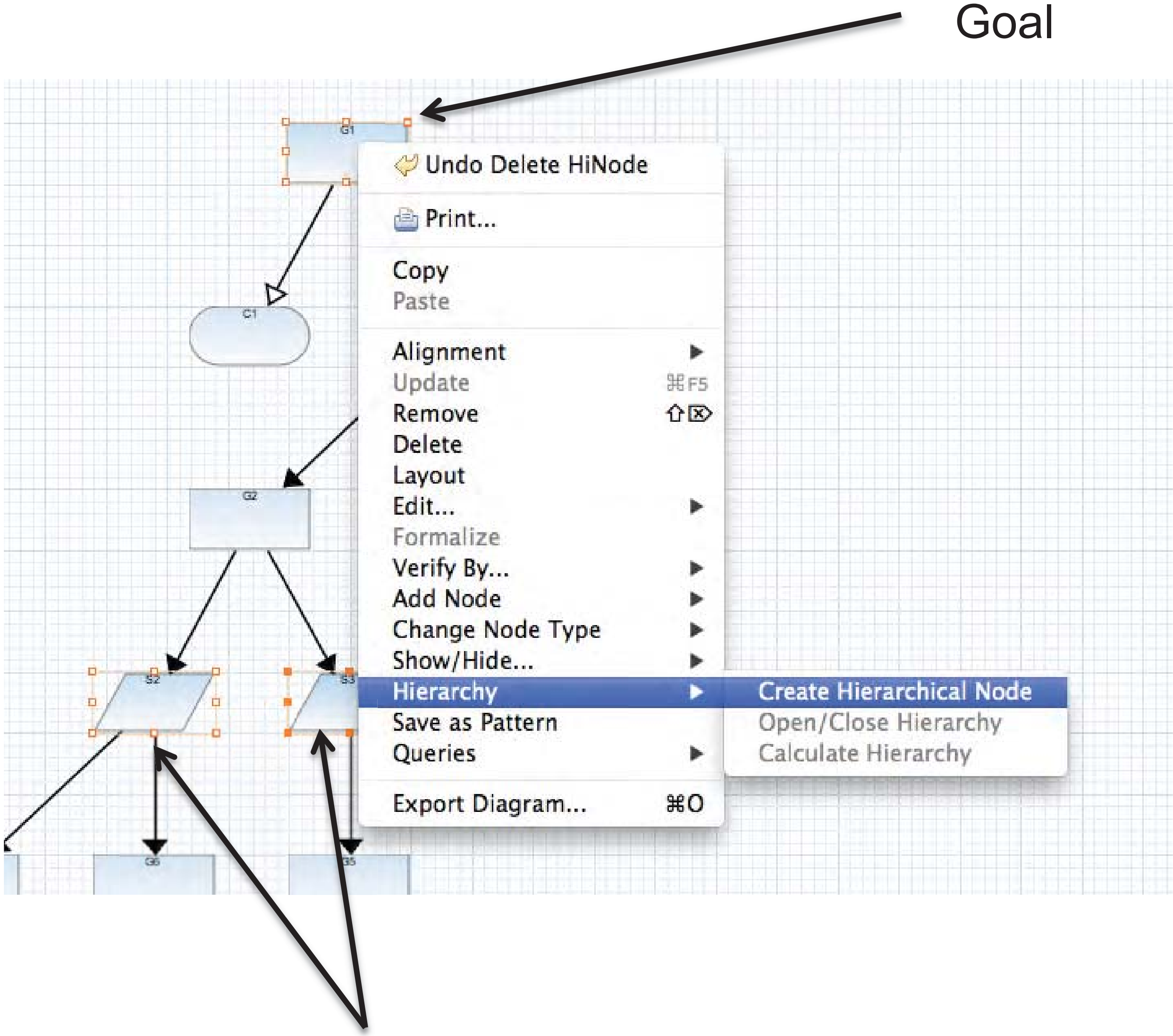


Example: Creating Hierarchical Goals

To create a hierarchical goal,

1. Select G1 as the root node.
2. Select S3 and S2 as the delimiting nodes for the hierarchy.
3. Right-click on the root node. Select Hierarchy→ Create Hierarchical Node

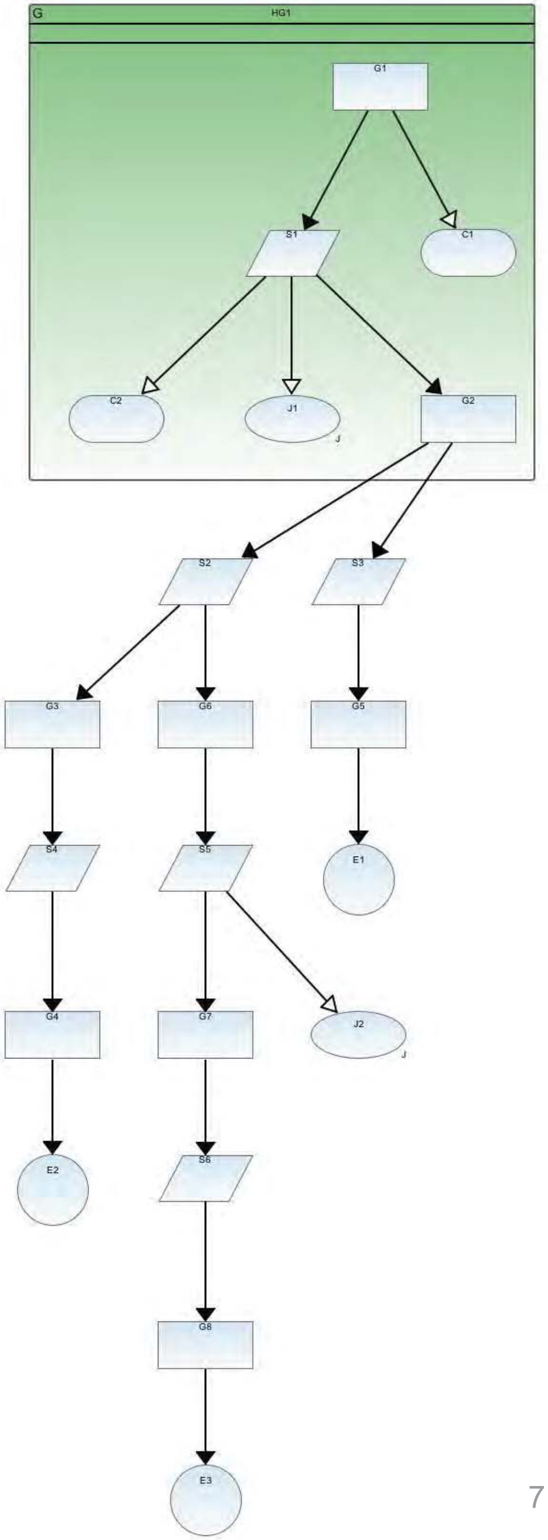
Root node: Goal



Delimiting nodes

Example: Creating Hierarchical Goals

The selected area is condensed into a hierarchy, leaving the rest of the original diagram intact.

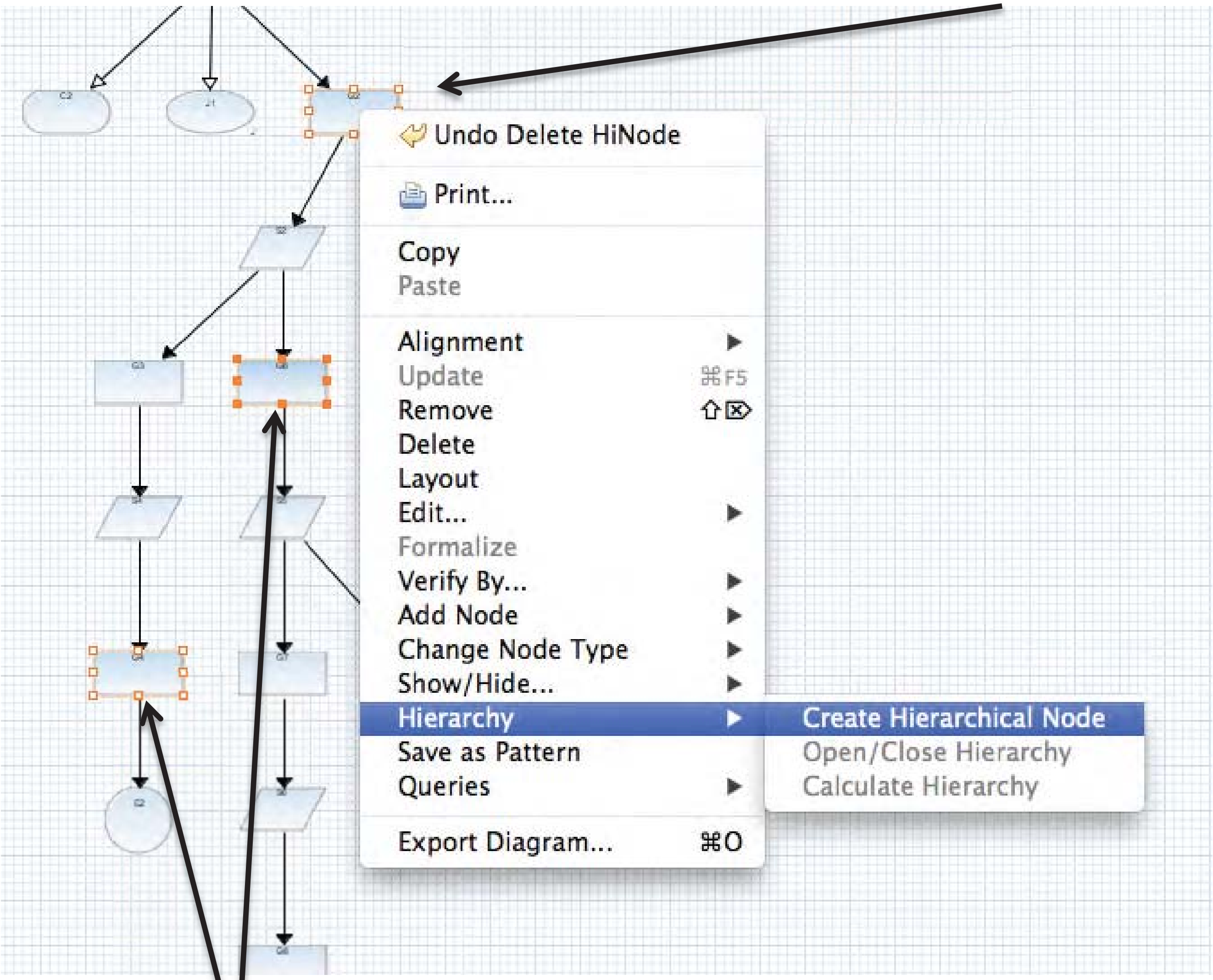


Hierarchical Goal

Example 1: Creating Hierarchical Strategies

To create a hierarchical strategy structure,

1. Select S2 as the root node.
2. Select G4 and G6 as the delimiting nodes for the hierarchy.
3. Right-click on the root node. Select Hierarchy→ Create Hierarchical Node

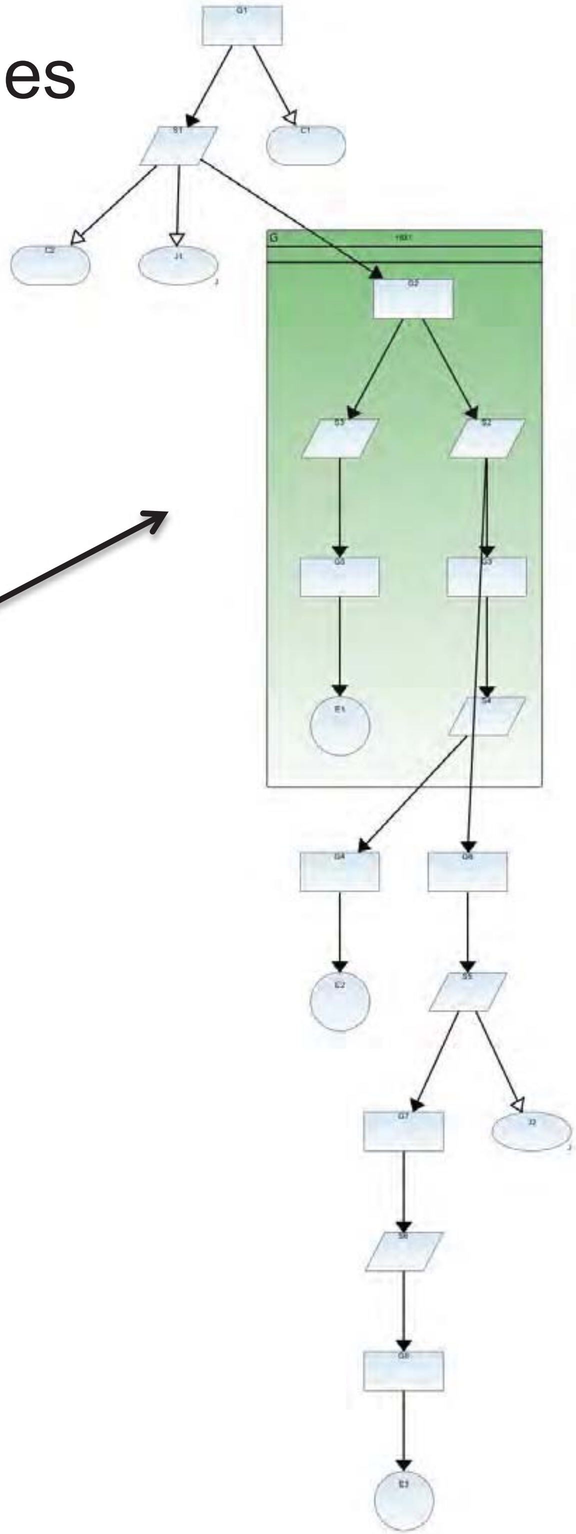


Root node: Strategy

Delimiting nodes

Example 1: Creating Hierarchical Strategies

The selected area is condensed into a hierarchy, leaving the rest of the original diagram intact.

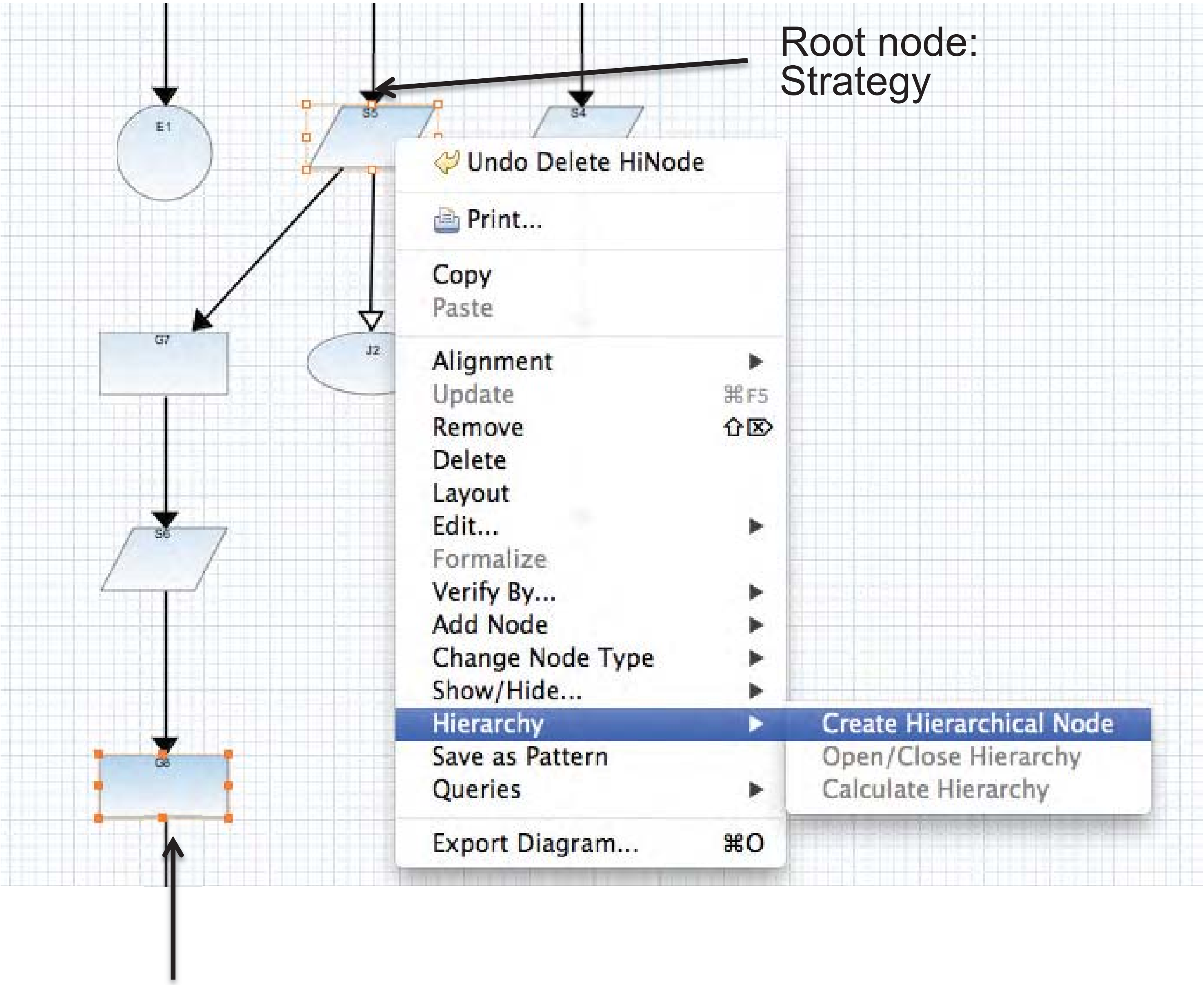


Hierarchical Strategy

Example 2: Creating Hierarchical Strategies Structures

To create another hierarchical strategy structure,

1. Select S5 as the root node.
2. Select G8 as the delimiting node for the hierarchy.
3. Right-click on the root node. Select Hierarchy→ Create Hierarchical Node

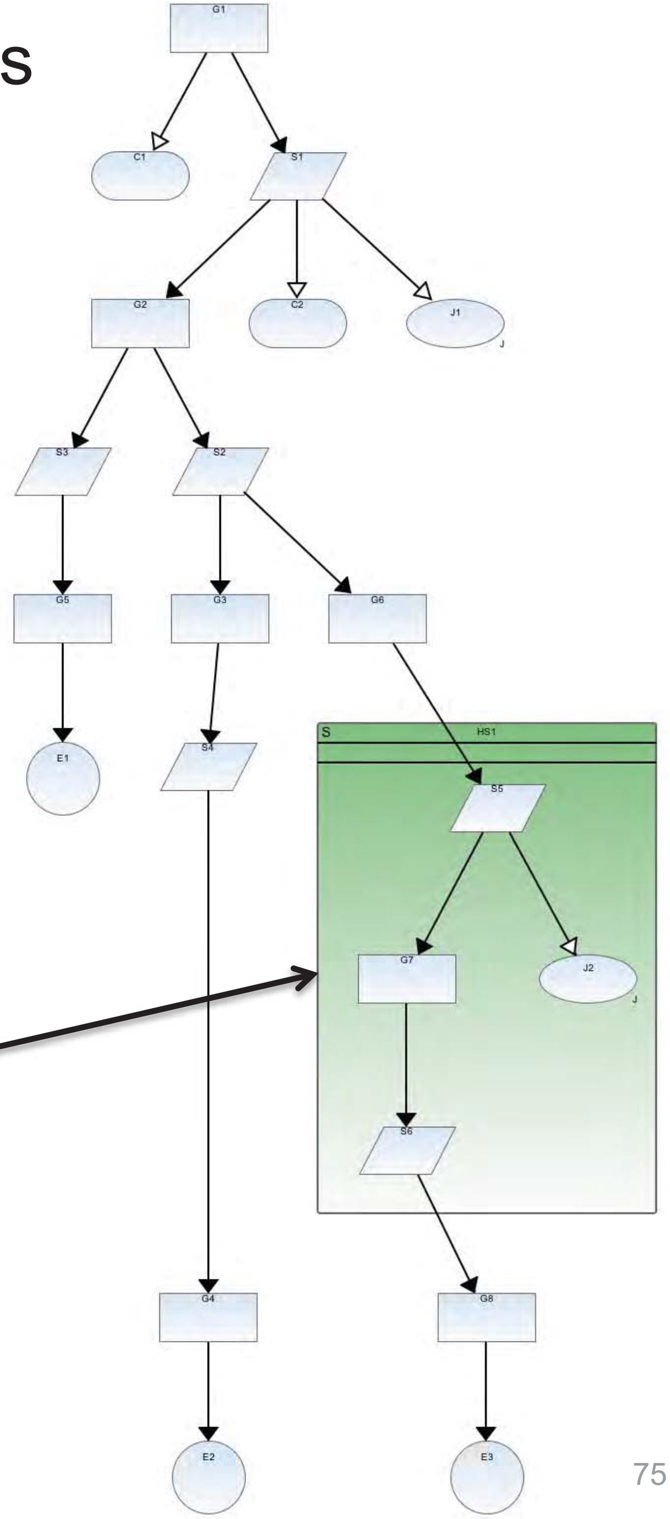


Root node: Strategy

Delimiting nodes

Example 2: Creating Hierarchical Strategies

The selected area is condensed into a hierarchy, leaving the rest of the original diagram intact.



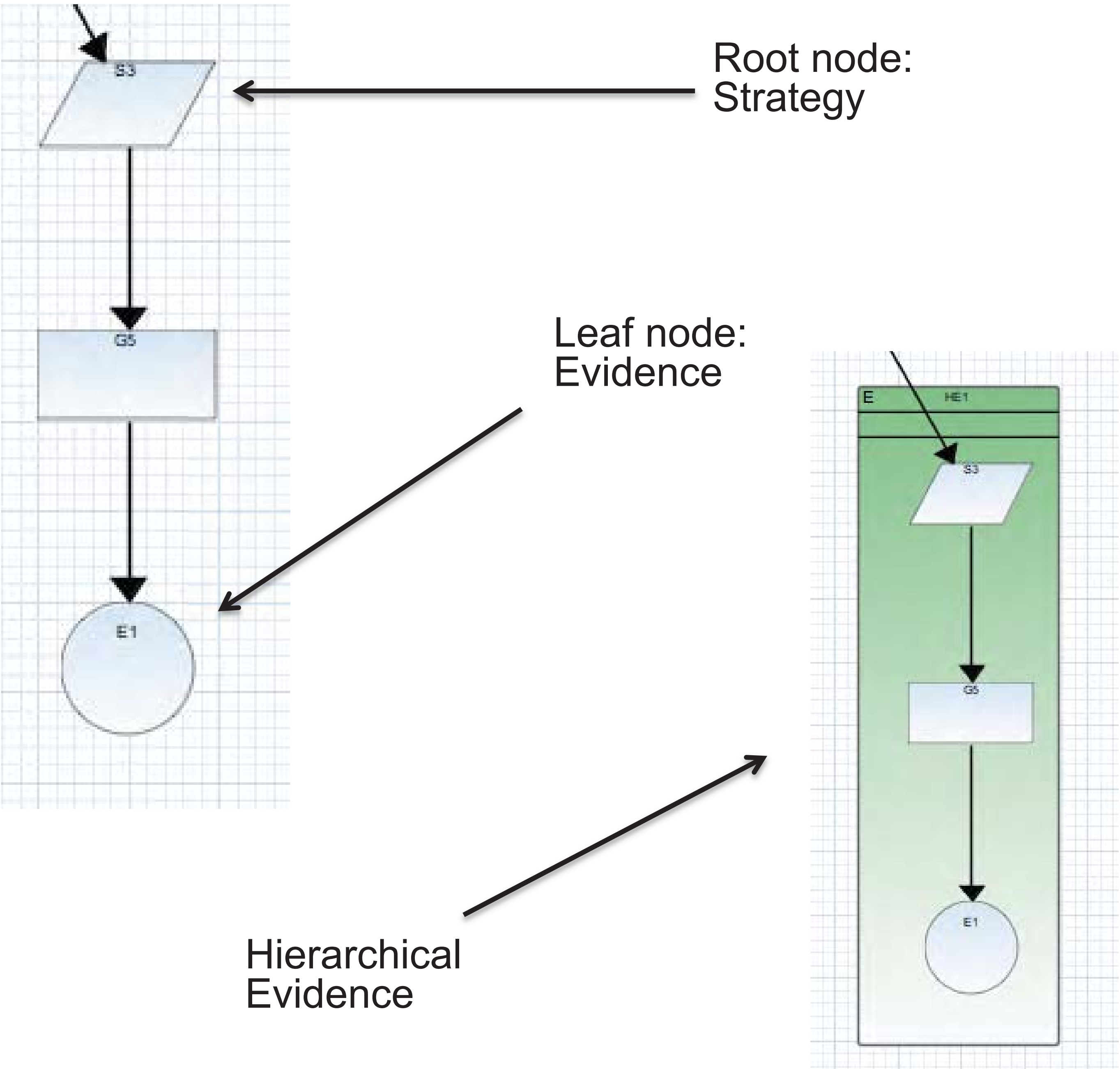
Hierarchical Strategy

Example: Creating Hierarchical Evidence

To create another hierarchical evidence structure,

1. Select S3 as the root node.
2. Right-click on the root node. Select Hierarchy→ Create Hierarchical Node

Note that because no delimiting nodes were selected, the hierarchy will include all nodes down to the leaf nodes.



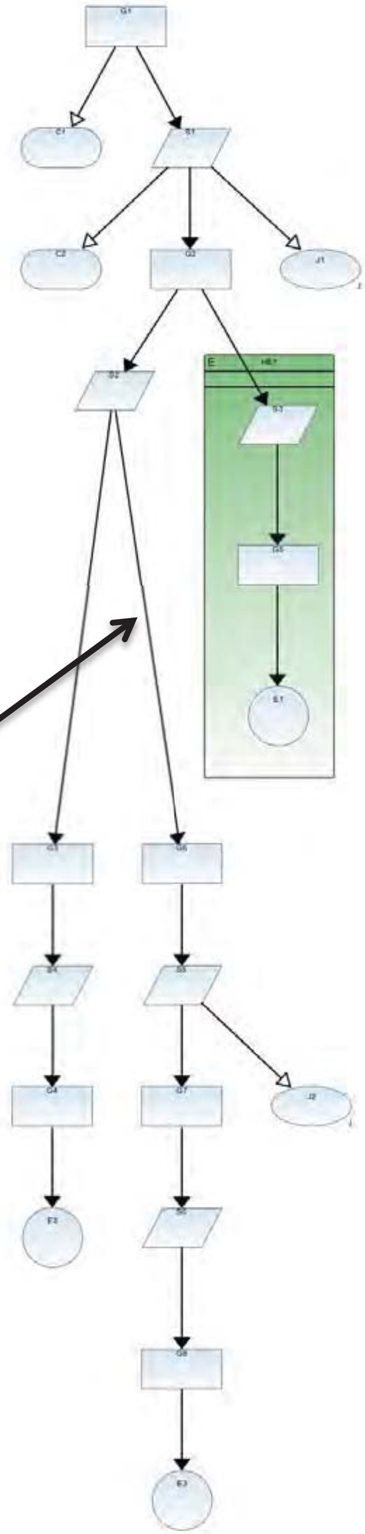
Root node: Strategy

Leaf node: Evidence

Hierarchical Evidence

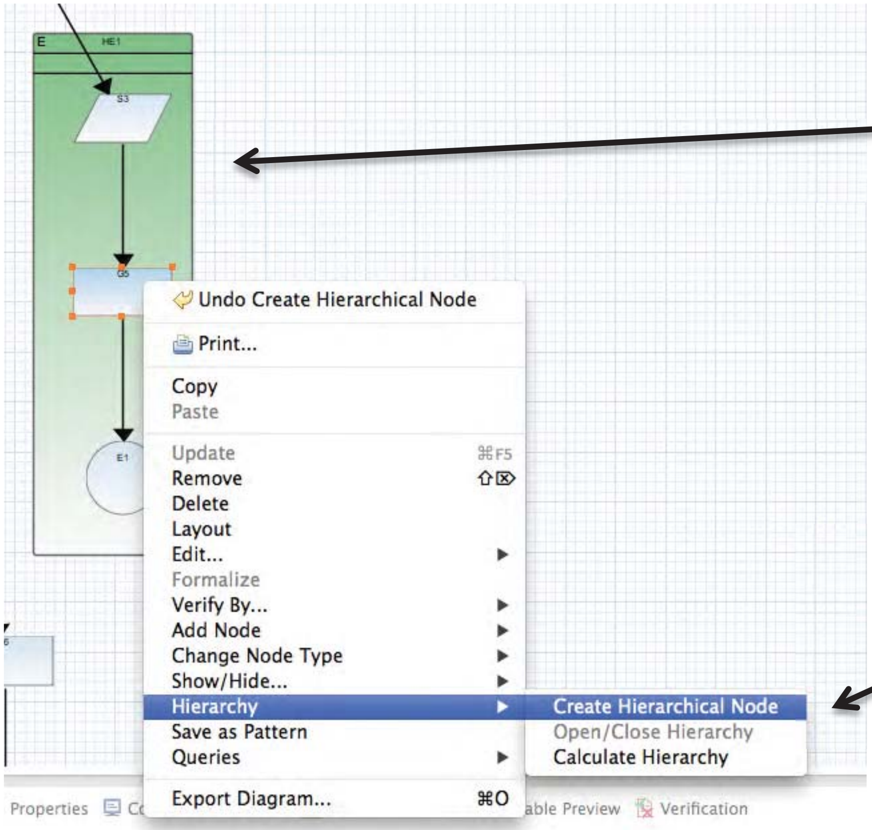
Example: Creating Hierarchical Evidence

The selected area is condensed into a hierarchy, leaving the rest of the original diagram intact.



Hierarchical Evidence

Example: Creating Nesting Hierarchies

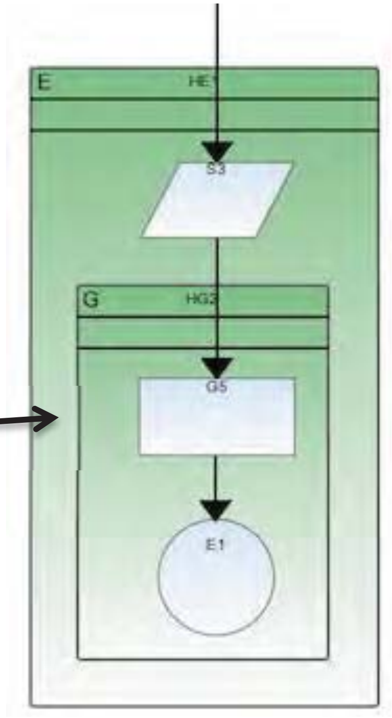


Outer Hierarchy

To create nested hierarchical structures (a hierarchy within a hierarchy):

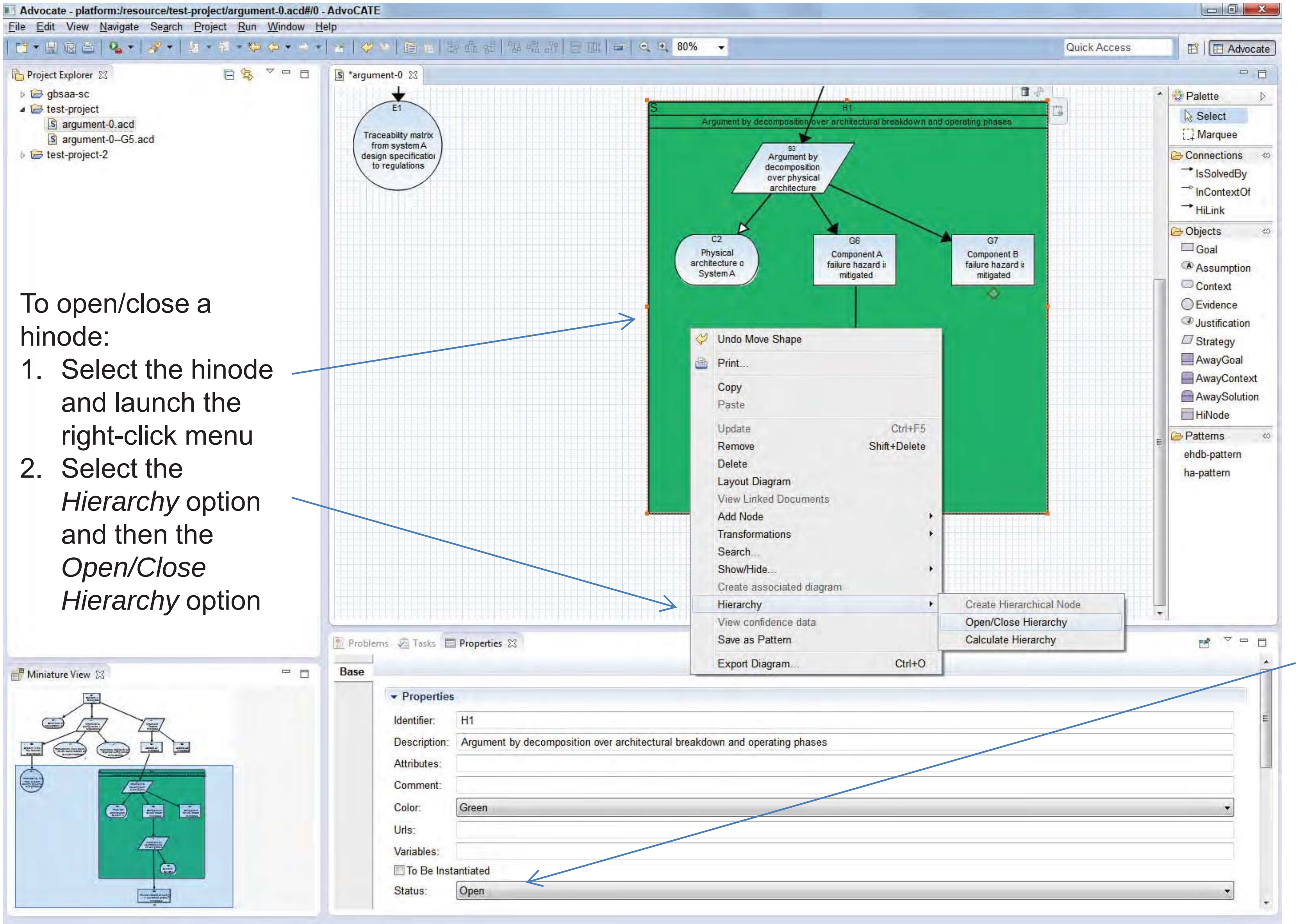
1. Within a hierarchy, select the a root node for the inner hierarchy.
2. Right-click on the root node. Select Hierarchy→ Create Hierarchical Node

Hierarchical Goal nested within a Hierarchical Evidence Structure



Opening/Closing Hierarchies

Hinodes can be closed/opened to collapse/expand the argument structure.



Identifier: H1

Description: Argument by decomposition over architectural breakdown and operating phases

Attributes:

Comment:

Color: Green

Urls:

Variables:

☐ To Be Instantiated

Status: Open

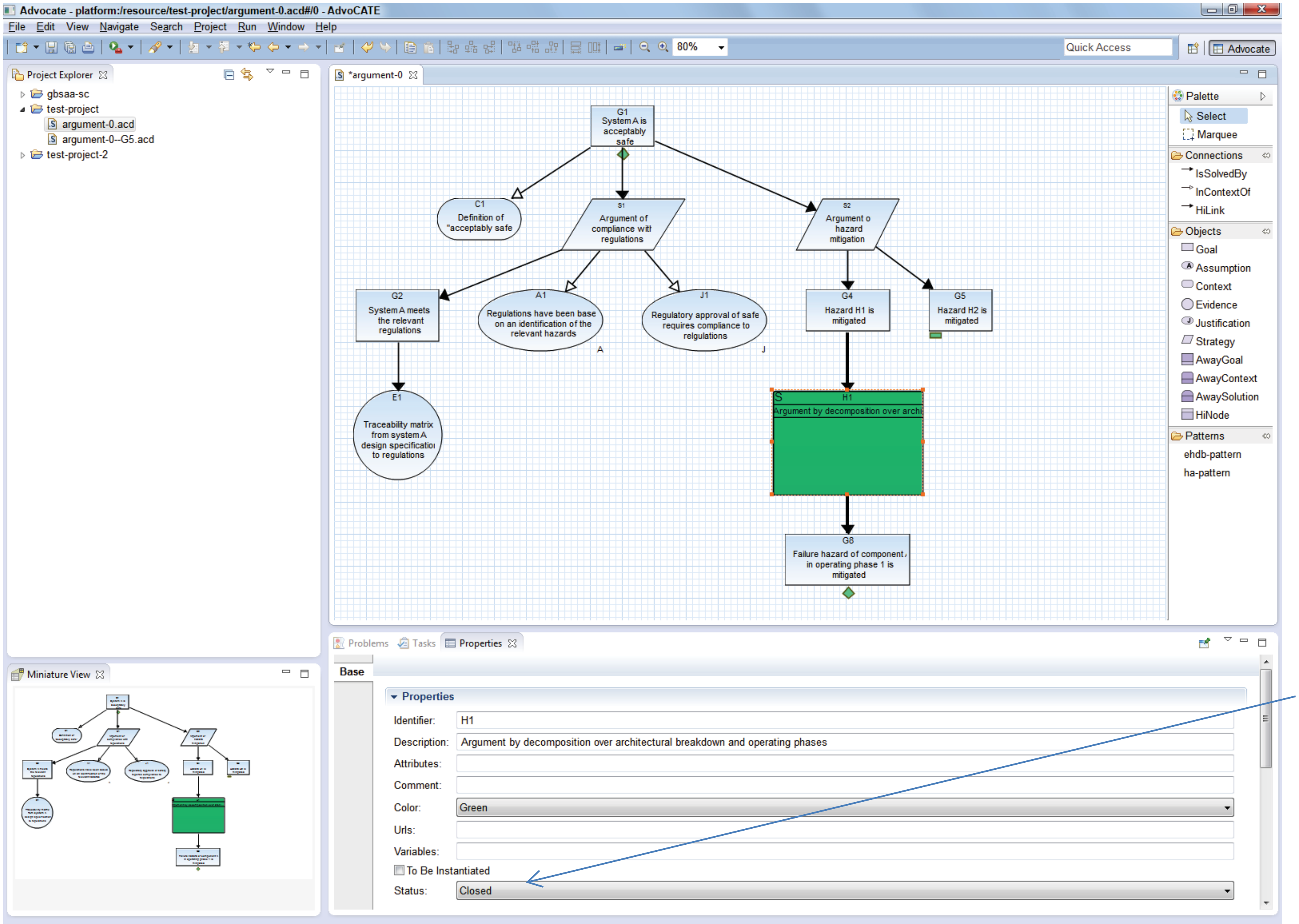
To open/close a hinode:

1. Select the hinode and launch the right-click menu
2. Select the *Hierarchy* option and then the *Open/Close Hierarchy* option

Hinodes have a default *Open* status

Opening/Closing Hierarchies

Closing a hinode collapses the node, improving visual comprehension by hiding its internal structure


Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Project Run Window Help
80%
Quick Access
Advocate
Project Explorer
gbsaa-sc
test-project
argument-0.acd
argument-0--G5.acd
test-project-2
*argument-0
G1
System A is acceptably safe
C1
Definition of "acceptably safe
S1
Argument of compliance with regulations
S2
Argument o hazard mitigation
G2
System A meets the relevant regulations
A1
Regulations have been base on an identification of the relevant hazards
A
J1
Regulatory approval of safe requires compliance to relgulations
J
G4
Hazard H1 is mitigated
G5
Hazard H2 is mitigated
E1
Traceability matrix from system A design specificatioi to regulations
S
H1
Argument by decomposition over archi
G8
Failure hazard of component in operating phase 1 is mitigated
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ehdb-pattern
ha-pattern
Miniature View
Problems
Tasks
Properties
Base

Properties

Identifier: H1

Description: Argument by decomposition over architectural breakdown and operating phases

Attributes:

Comment:

Color: Green

Urls:

Variables:

□ To Be Instantiated

Status: Closed

Hinodes status is updated to *Closed* on closing the hierarchy

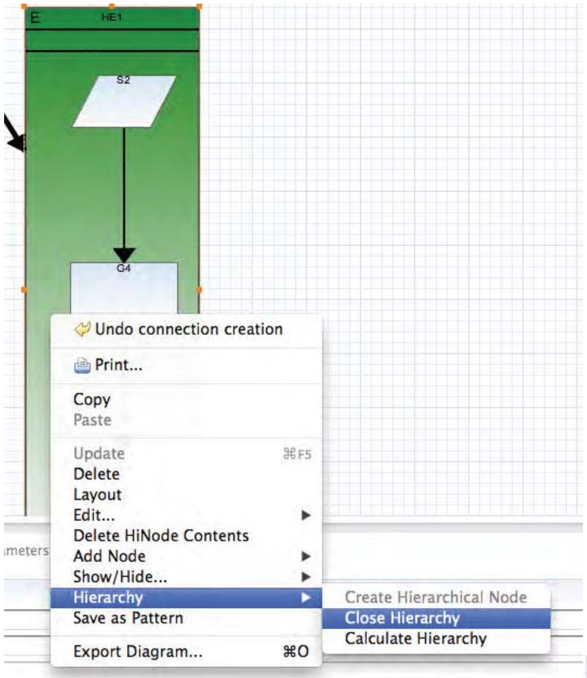
Example: Closing a Hierarchy

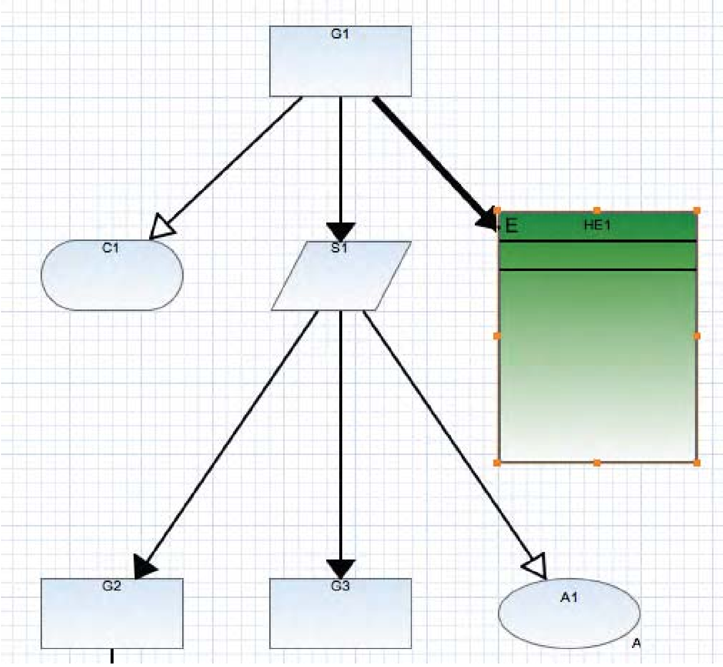
Closing a hierarchy will minimize the hierarchical structure, preserving the internal nodes.

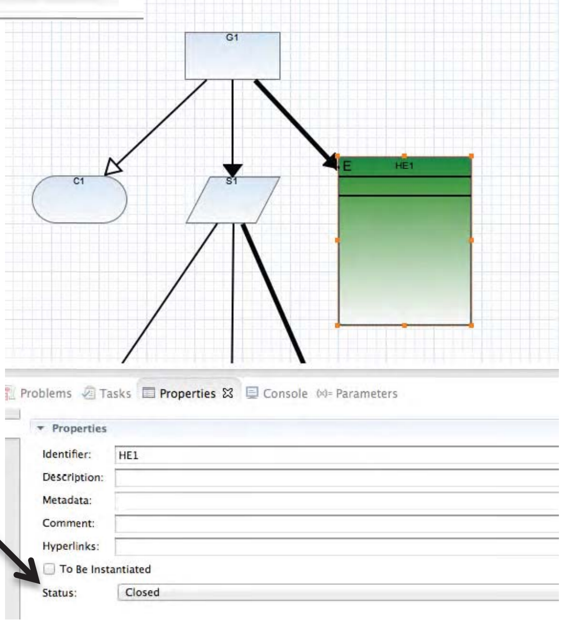
To close the hierarchy,

1. Click on the hierarchy.
2. Select Hierarchy→ Close Hierarchy. In the hierarchy Properties, the status will indicate that the hierarchy is Closed.

Deleting the hierarchy will delete the hierarchical structure but not its contents. However, selecting Delete HiNode Contents will delete the nodes contained within the hierarchy.







Properties

Identifier: HE1

Description:

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

Status: Closed

Calculating Hierarchies

- Calculating hierarchy screenshot

Modules

Modules prevent complicated diagrams from becoming too unwieldy by providing connections between arguments and patterns contained on separate canvas tabs.

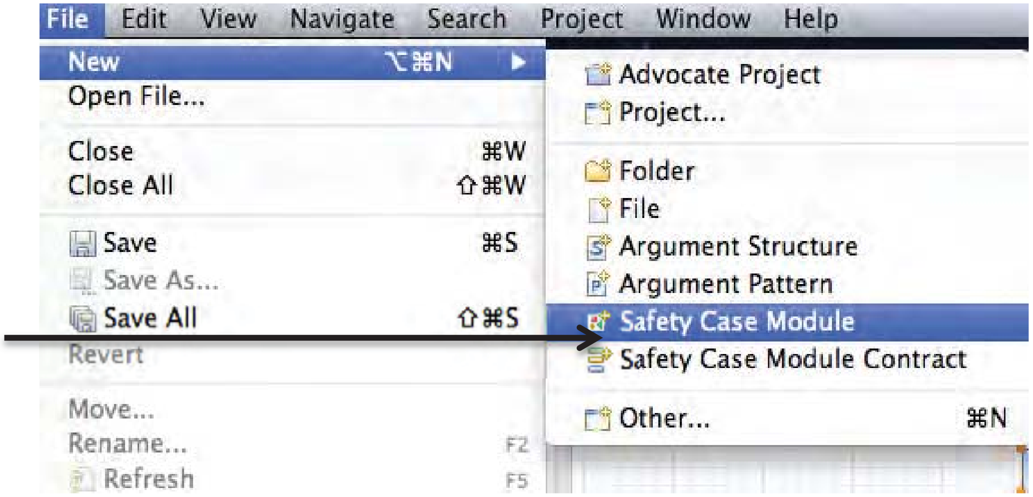
The two types of modules are:

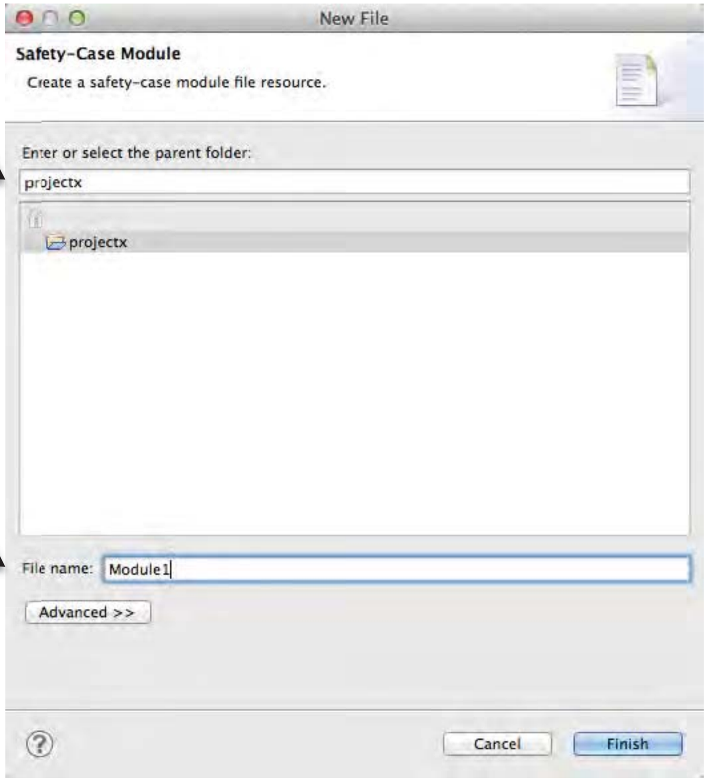
- Safety Case Module
- Safety Case Module Contract

Creating a Module

To create a module canvas on a new tab

1. Select File → New → Safety Case Module
2. Select the parent folder (usually a project) in which to put the module
3. Name the module
4. Click Finish




New File

Safety-Case Module

Create a safety-case module file resource.

Enter or select the parent folder: projectx

projectx

File name: Module1

Creating a Module (cont.)

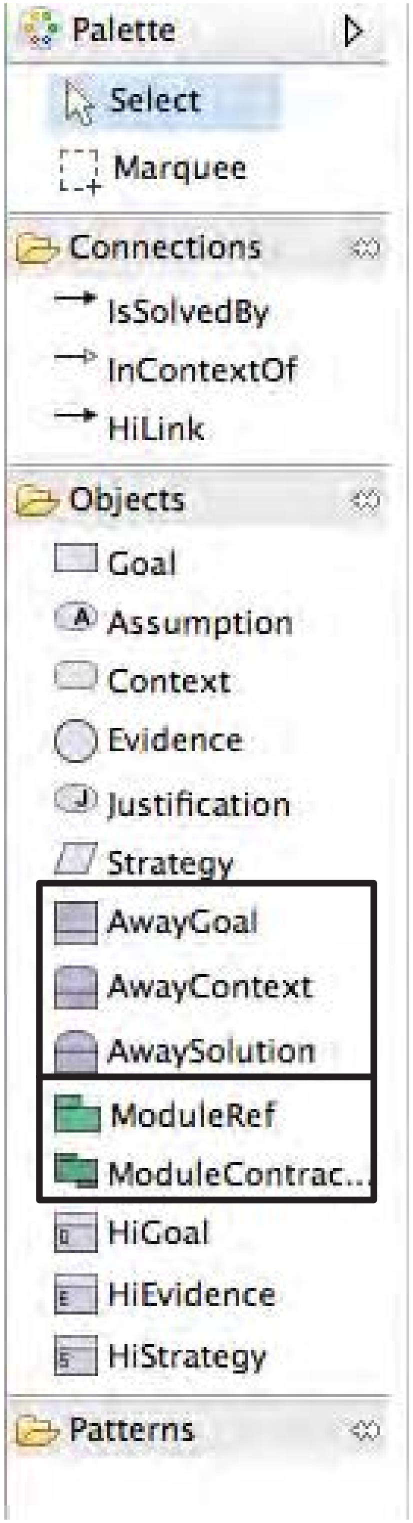
Notice that the new module tab contains an empty canvas as well as some new node types in the palette.

These node types link to goal, context, and solution/evidence nodes in other modules or arguments:

- Away Goal
- Away Context
- Away Solution

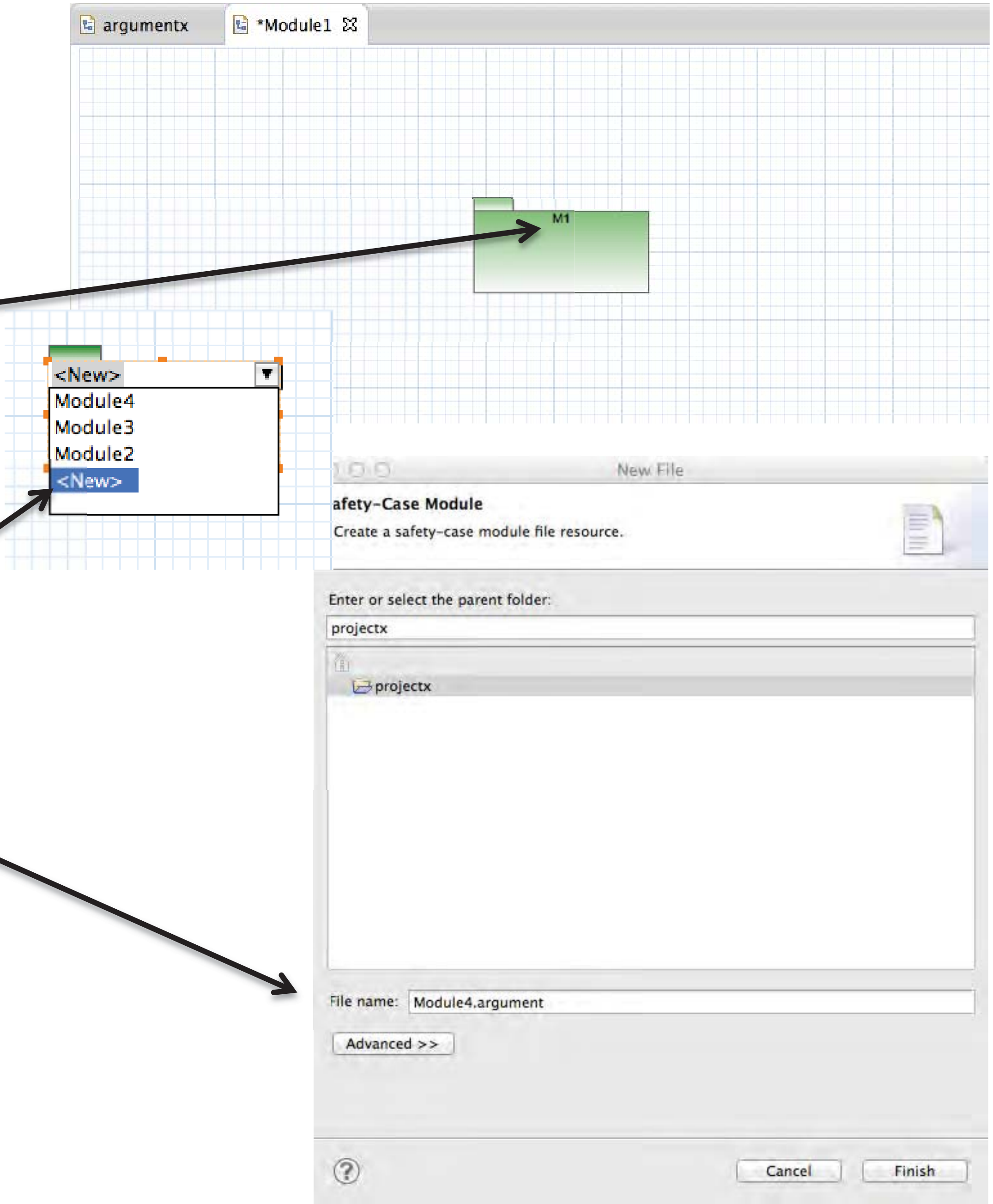
These node types allow link to other modules or module contract references:

- Module Reference—link to other modules
- Module Contract Reference—link to other module contract references



Example: Linking an existing module reference to a new (non-existing) module

- Add a ModuleRef object by dragging and dropping or using the shortcut
- Select the ModuleRef object and click on the text, which will display a dropdown menu with all existing modules in that project
- To link to a new module that has not yet been created, select <New>
- In the New File dialog box, enter the name of the new module. (The parent folder will automatically default to the name of the project.) Click Finish.



afety-Case Module

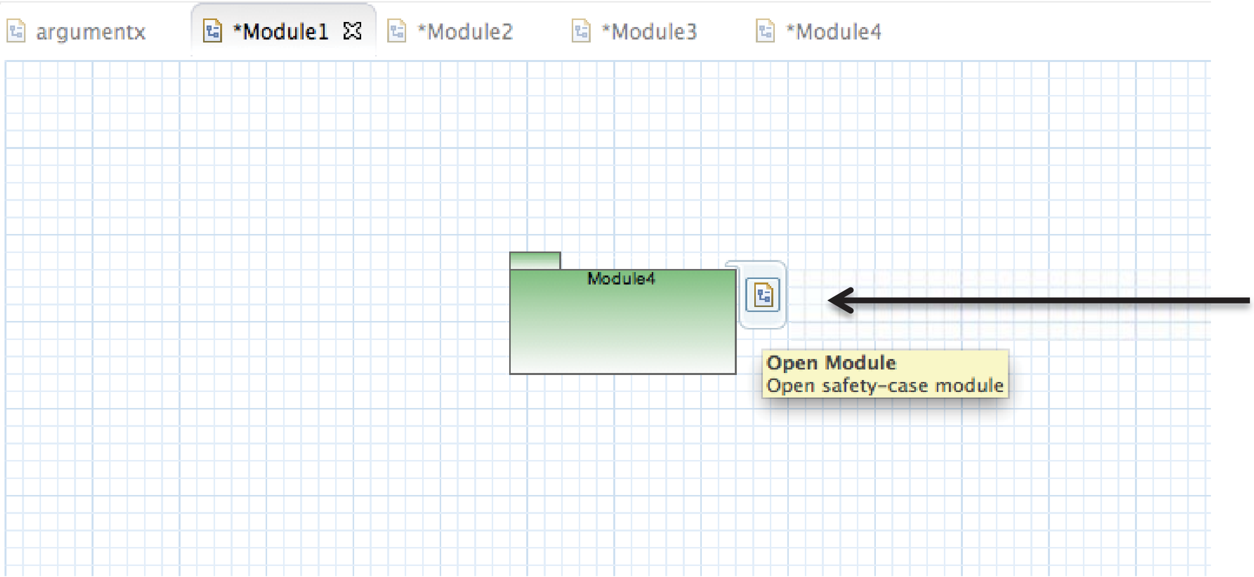
Create a safety-case module file resource.

Enter or select the parent folder: projectx

projectx

File name: Module4.argument

Example (cont.): Linking an existing module reference to a new (non-existing) module

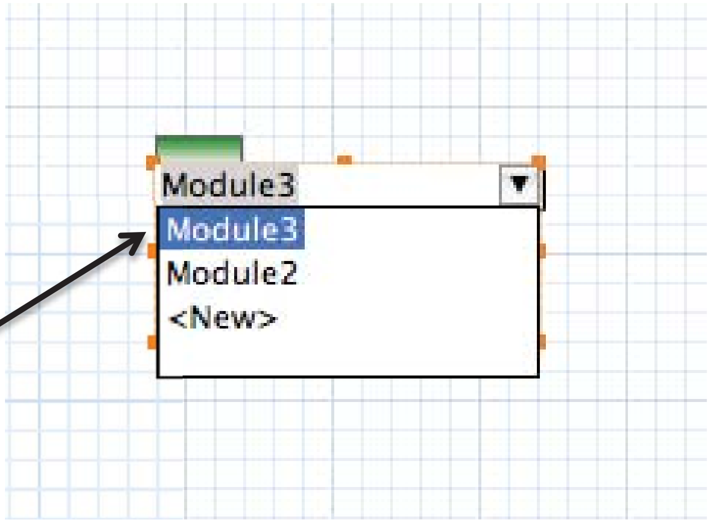


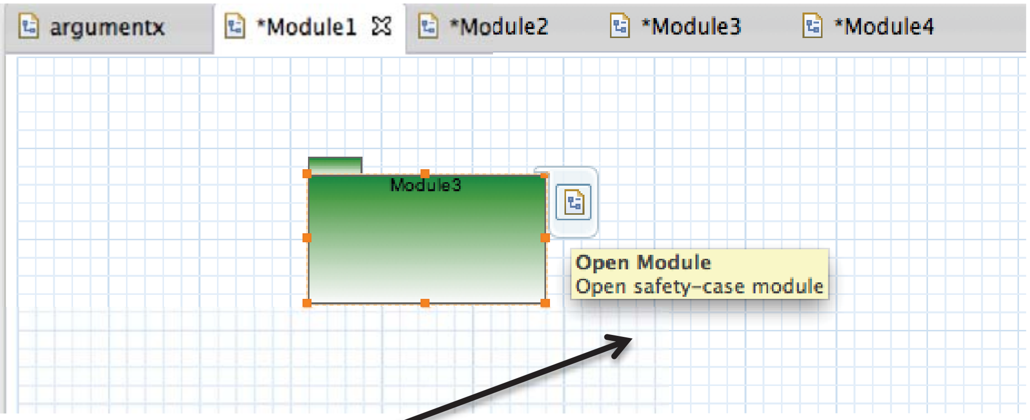
Open Module
Open safety-case module

- The ModuleRef object will be renamed to the name of the target module—in this case, Module4.
- To open the target module, click on the icon on the upper right portion of the ModuleRef object and select Open Module. Clicking this link will open the target Module canvas.

Example: Linking an existing module reference to an existing module

- Select the ModuleRef object and click on the text, which will display a dropdown menu with all existing modules in that project.
- Select the target module to link that object to.
- The ModuleRef object will be renamed to the name of the target module—in this case, Module3.
- To open the target module, click on the icon on the upper right portion of the ModuleRef object and select Open Module. Clicking this link will open the target Module canvas.



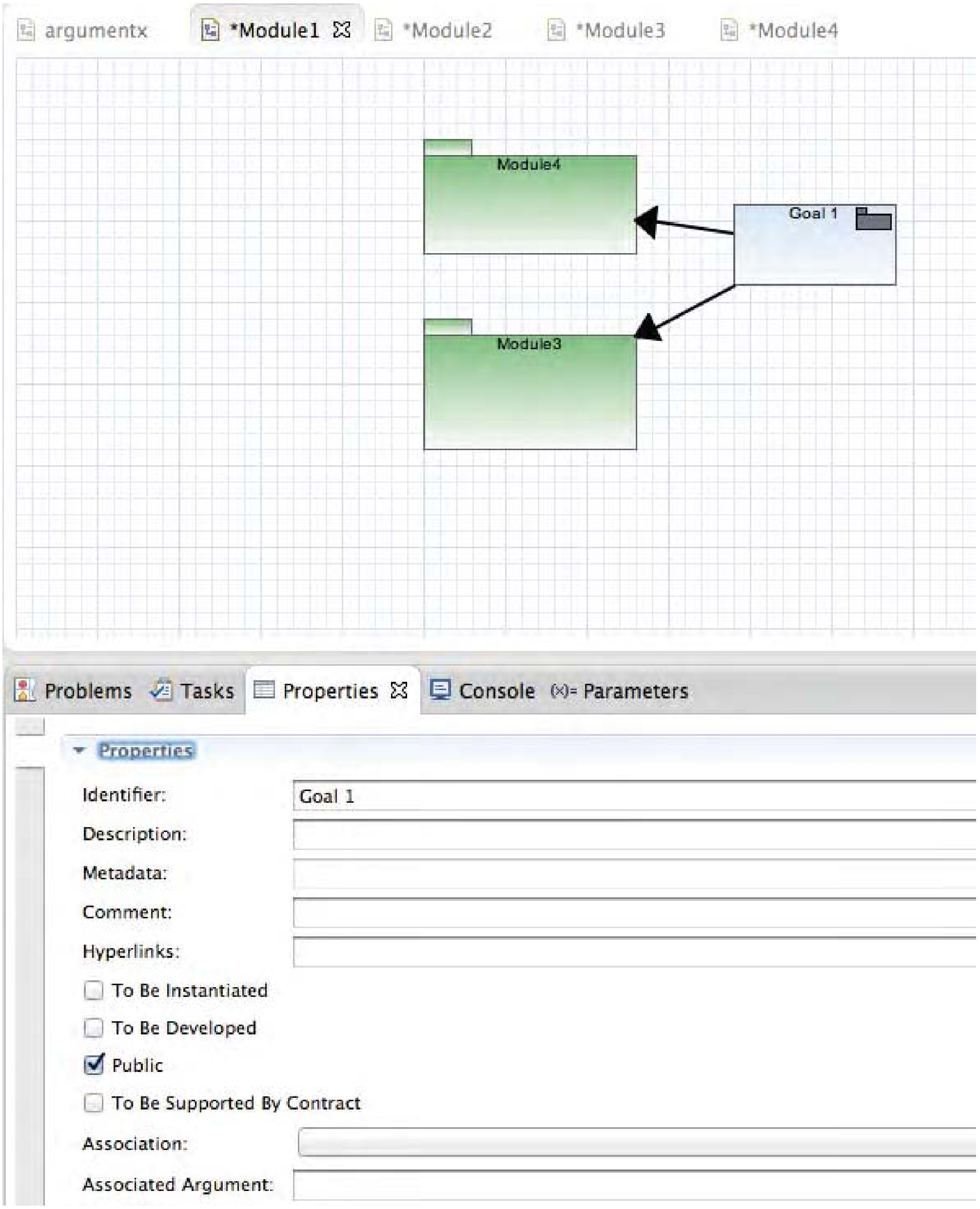


Example: Away Nodes

Away nodes simplify complicated diagrams and link between nodes in separate modules, arguments, or structures.

Any target node must be marked “public” so that it is viewable when linked from other modules.

Add a node to the canvas—in this case, a goal node called Goal 1—and in its Properties, check the Public box. Notice the folder icon that appears in the top right of the node, indicating that it’s “publicly” viewable.



Identifier: Goal 1

Description:

Metadata:

Comment:

Hyperlinks:

☐ To Be Instantiated

☐ To Be Developed

☑ Public

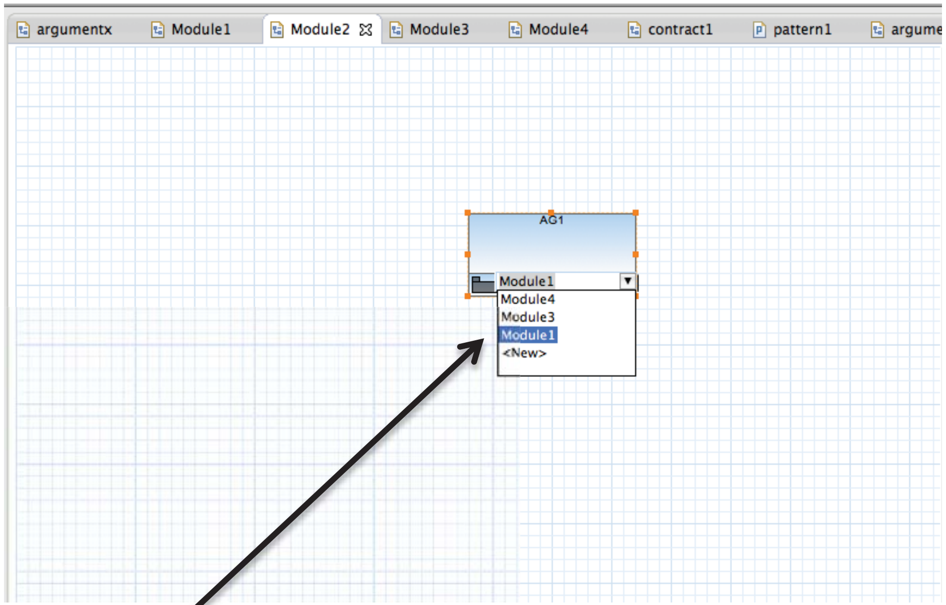
☐ To Be Supported By Contract

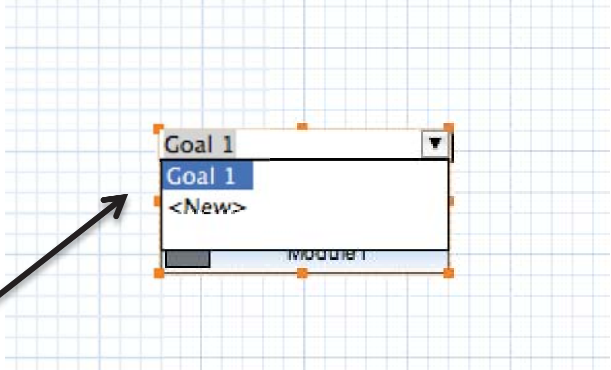
Association:

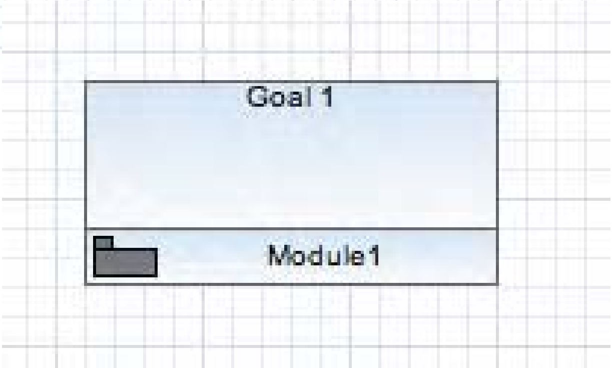
Associated Argument:

Example (cont.): Away Nodes

- In the source module, add an away node—in this case, add an AwayGoal to the canvas of Module2. This node can be linked to any target goal in another module. In this example, AG1 will be linked to Goal 1 in Module1. Notice the folder icon that appears in the bottom rectangle on the AG1 node.
- Click the drop-down menu beside the folder icon and select the target module to reference (the one that the target node is in)
- Click the away node ID and select the node to link it to from the drop-down menu, which will only display target nodes of the same type as the source away node that within the reference module.



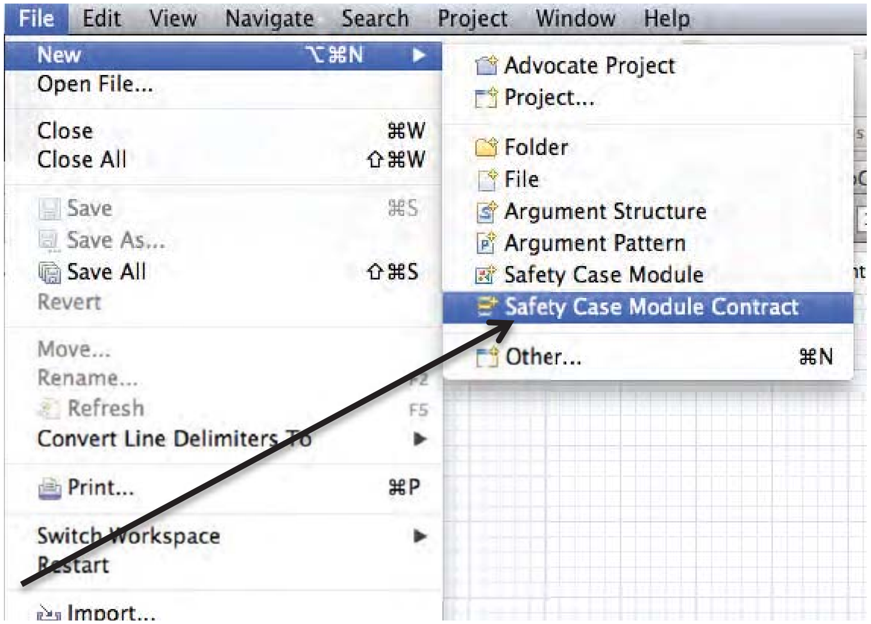


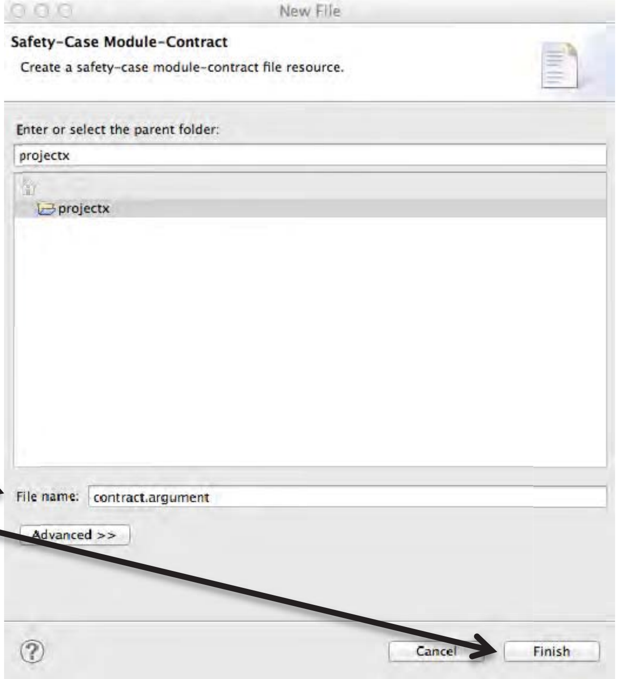


Creating Module Contracts

- Module contracts serve as a justification for how two modules reference one another; rather, how one module makes certain assumptions that are supported by the guarantees implicit by another module.
- To create a Module Contract,

1. Go to File→New→ Safety Case Module Contract.
2. Select the parent folder (usually a project) in which to put the module.
3. Name the module.
4. Click Finish.





Safety-Case Module-Contract

Create a safety-case module-contract file resource.

Enter or select the parent folder: projectx

projectx

File name: contract.argument

Creating Module Contracts (cont.)

As with creating modules, notice that the new module tab contains an empty canvas as well as some new node types in the palette.

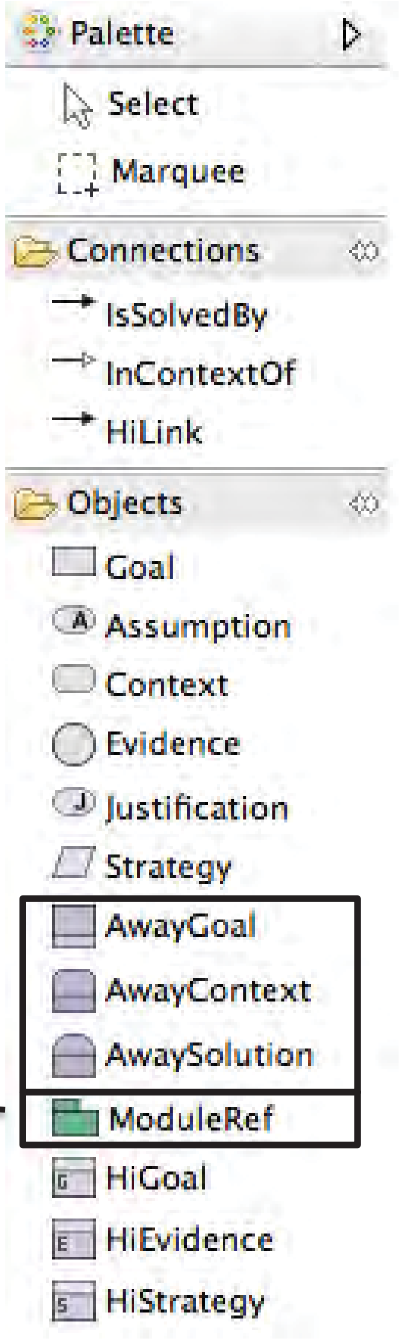
These objects allows linkages between nodes in other modules, arguments, or structures:

- AwayGoal
- AwayContext
- AwaySolution

This object allow linkages between other modules:

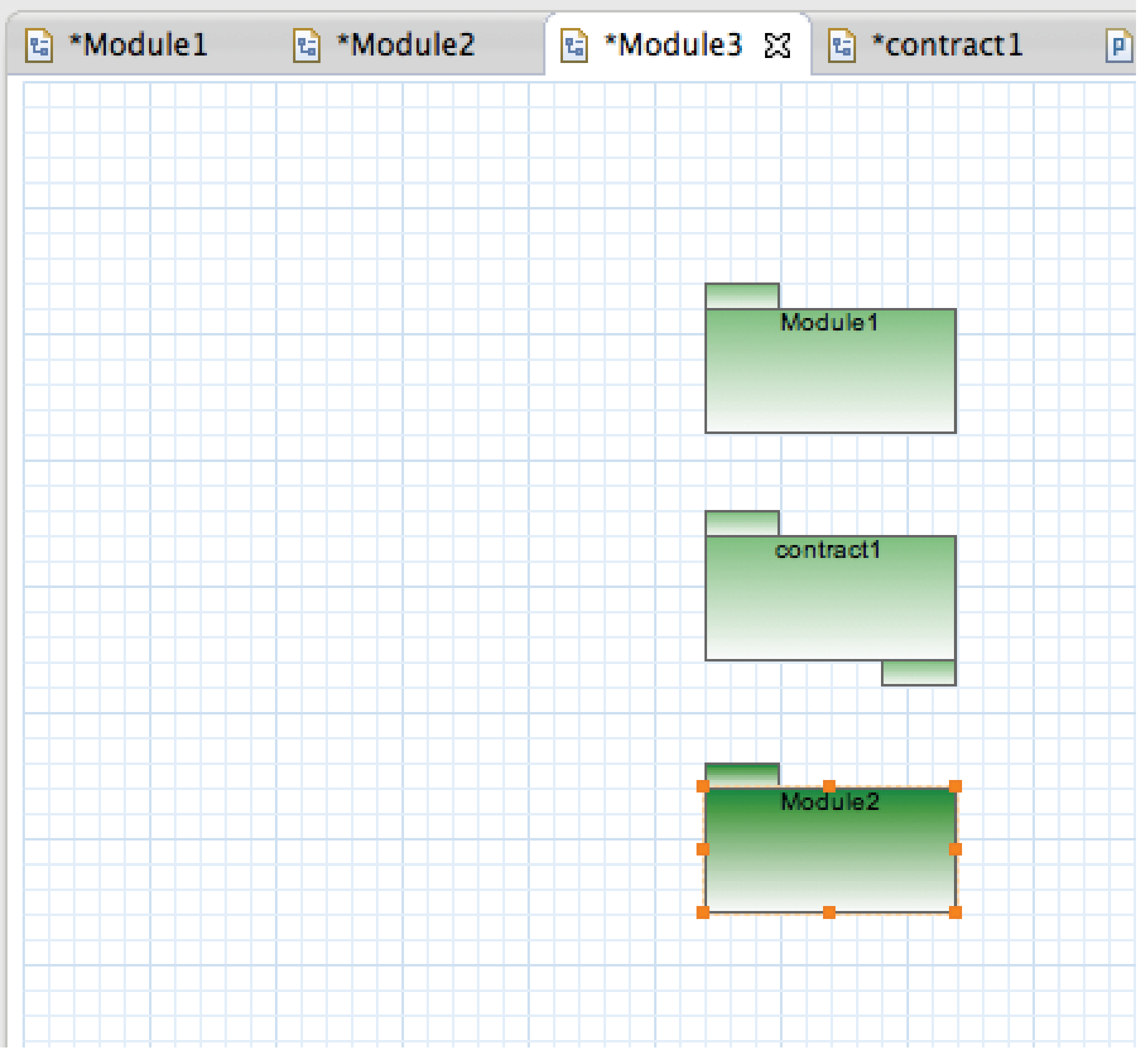
- ModuleRef

Notice that when creating a module contract diagram, other module contracts cannot be referenced.



Example: Using Module Contracts in Diagrams

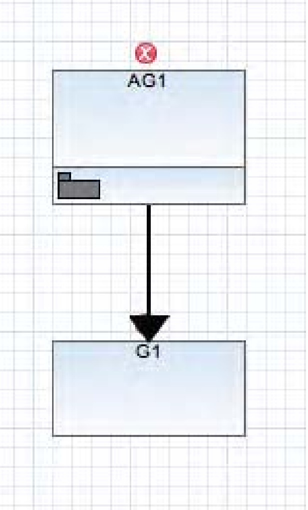
- Like Modules, Module Contracts can be used to simplify complicated diagrams.
- The Module Contract Reference object acts as a placeholder for Module Contracts within other modules.
- See the example at right, illustrating the relationships between several Modules and Module Contracts with Module or Module Contract reference objects.

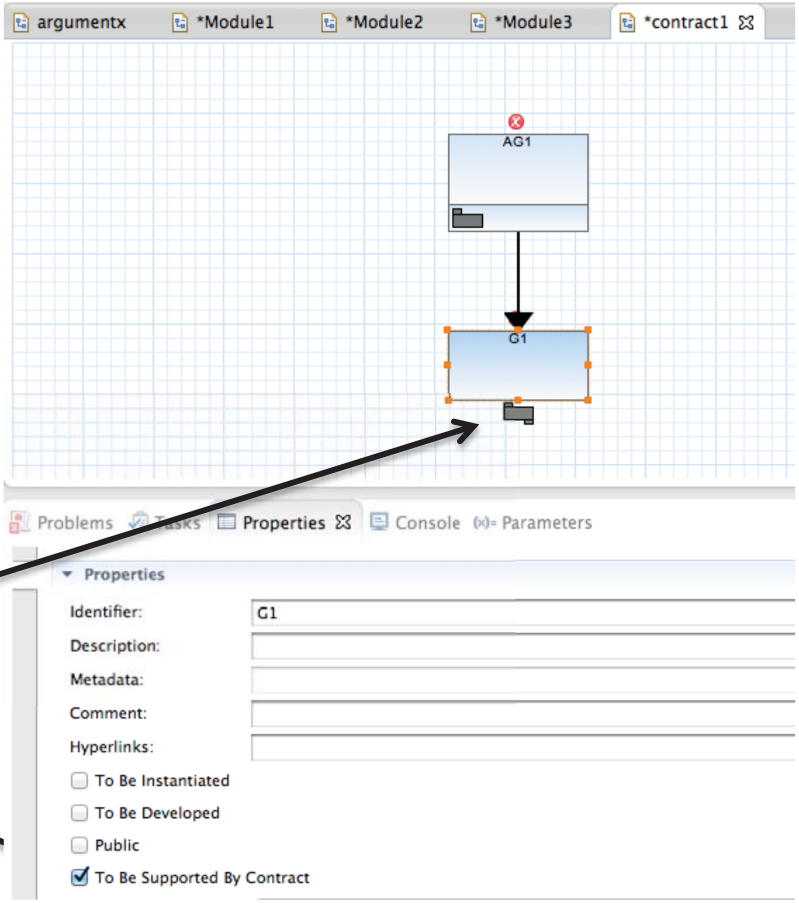


Example: Using Away Nodes in Module Contracts

- An away node may feature outgoing links within a module contract
- For example, to create an Away Goal,

1. Drag and drop the Away Goal object from the palette onto the canvas (or use the shortcut).
2. Drag and drop the Goal object from the palette onto the canvas (or use the shortcut). Add a link between the nodes.
3. Within the Goal node's Properties, check the box "To Be Supported by Contract". Notice that a module contract reference icon appears below the Goal node.





Properties

Identifier: G1

Description:

Metadata:

Comment:

Hyperlinks:

☐ To Be Instantiated

☐ To Be Developed

☐ Public

☑ To Be Supported By Contract

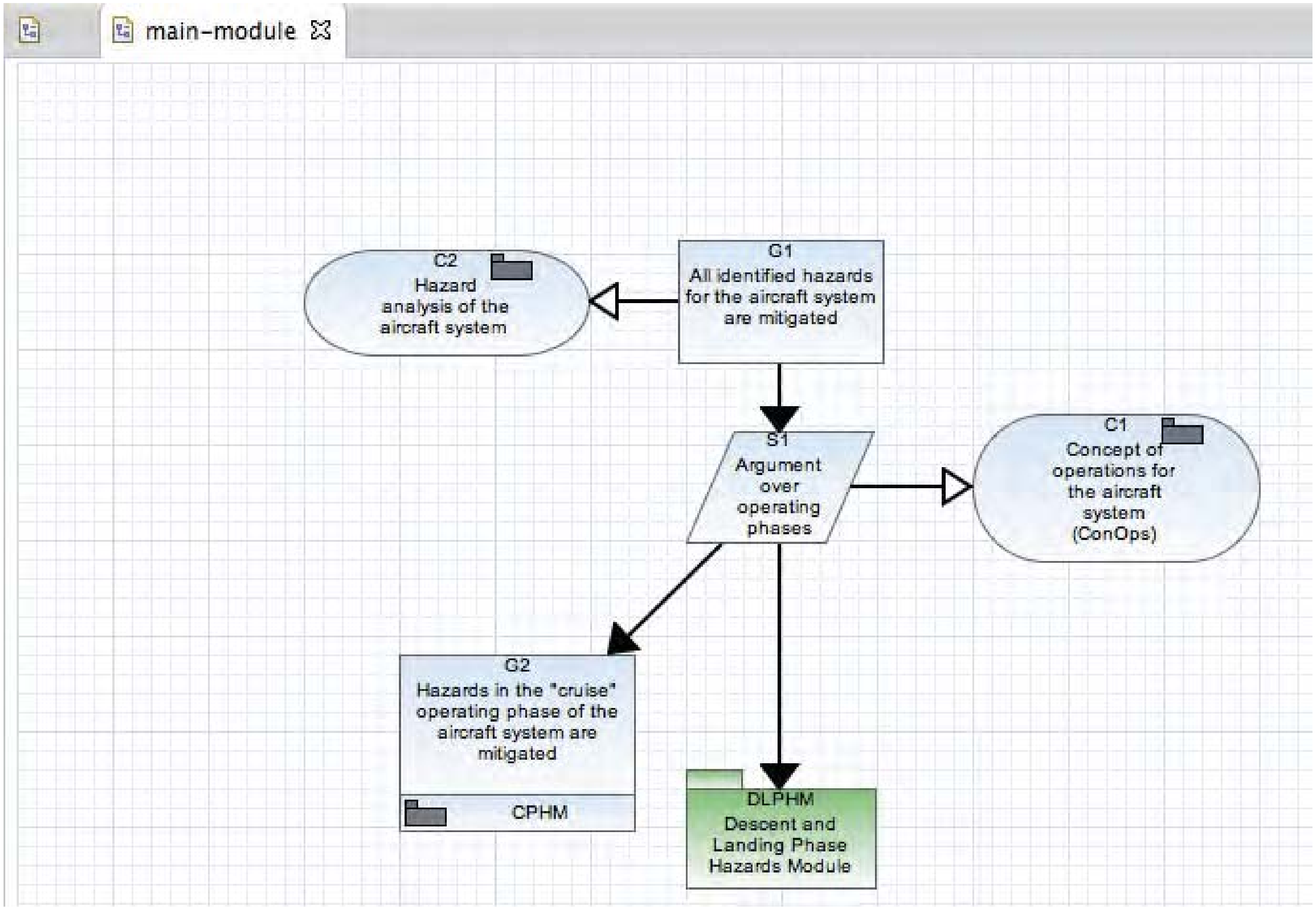
Putting it All Together: Complex Diagram Walkthrough

Here is an example of a more complex diagram by a hypothetical user that integrates the concepts introduced during previous slides.

The following slides will show, step by step, how to create the diagram shown here and link it to other modules in various ways.

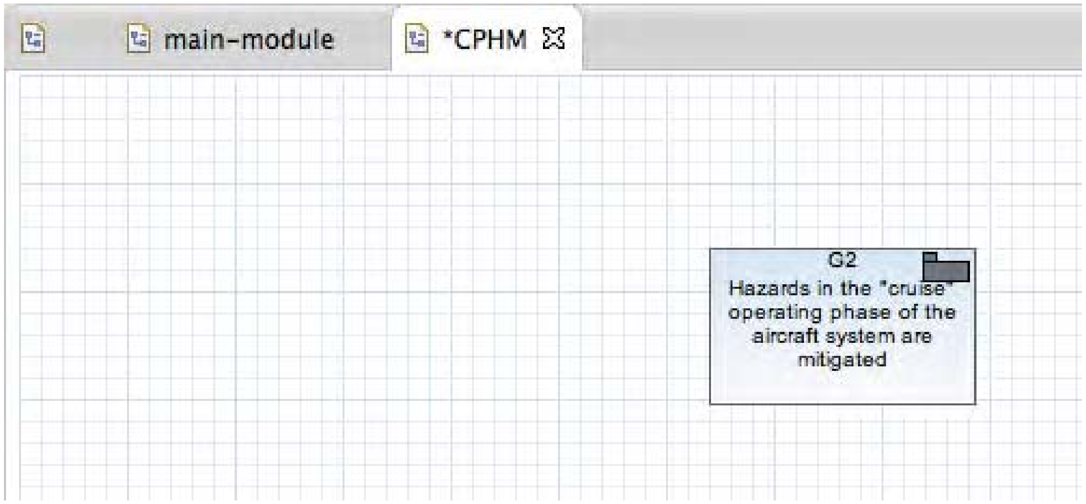
This user's module diagram consists of:

- A Goal G1
- A Context C2, marked public
- A Strategy S1
- A Context C1, marked public
- An Away Goal G2, linked to another module CPHM
- And a module reference DLPHM


main-module
C2
Hazard analysis of the aircraft system
G1
All identified hazards for the aircraft system are mitigated
S1
Argument over operating phases
C1
Concept of operations for the aircraft system (ConOps)
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
CPHM
DLPHM
Descent and Landing Phase Hazards Module

Putting it All Together: Complex Diagram Walkthrough (cont.)

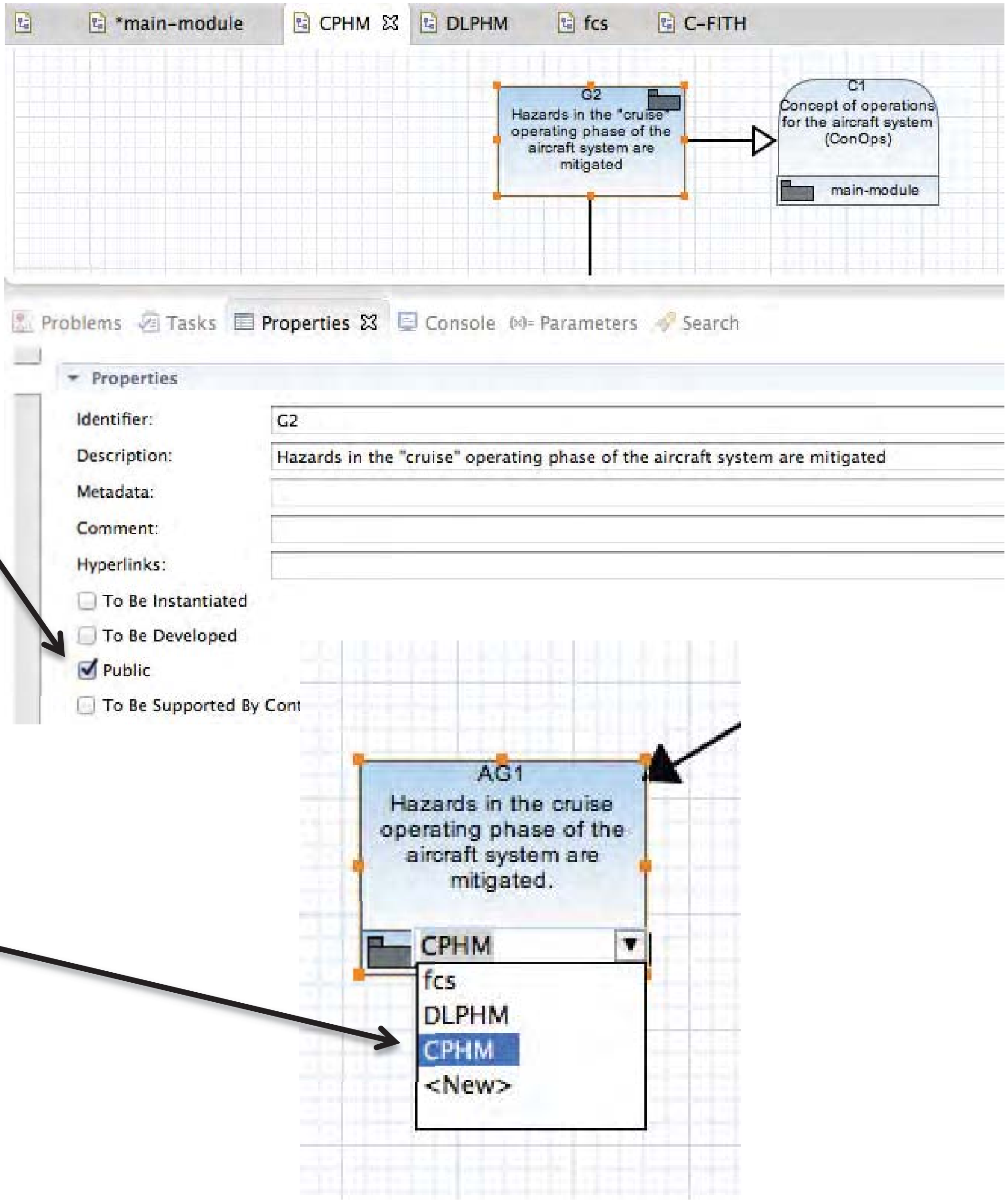
To create the main-module diagram, the start by employing the keyboard shortcuts to add nodes G1, S1, C1 and C2 and then fill in their descriptions.
To link this main-module argument to another module called CPHM, which has already been created, link the diagrams through a goal node with the description "Hazards in the 'cruise' operating phase of the aircraft are mitigated". This is G2 in CPHM.


main-module
*CPHM
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated

Putting it All Together: Complex Diagram Walkthrough (cont.)

To link the modules, the user:

1. Navigates to the Goal G2 in CPHM and marks it public in the node's Properties.
2. Next, the user adds an Away Goal to the diagram and fills in its description. Notice at this point that the identifier of the away goal is AG1.
3. Next, the user specifies which module the Away Goal AG1 will link to. The menu lists all existing public modules in this project. The user selects, in this case, CPHM.



*main-module
CPHM
DLPHM
fcs
C-FITH
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
C1
Concept of operations for the aircraft system (ConOps)
main-module
Problems
Tasks
Properties
Console
Parameters
Search
AG1
Hazards in the cruise operating phase of the aircraft system are mitigated.
CPHM
fcs
DLPHM
CPHM
<New>

Properties

Identifier: G2

Description: Hazards in the "cruise" operating phase of the aircraft system are mitigated

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

□ To Be Developed

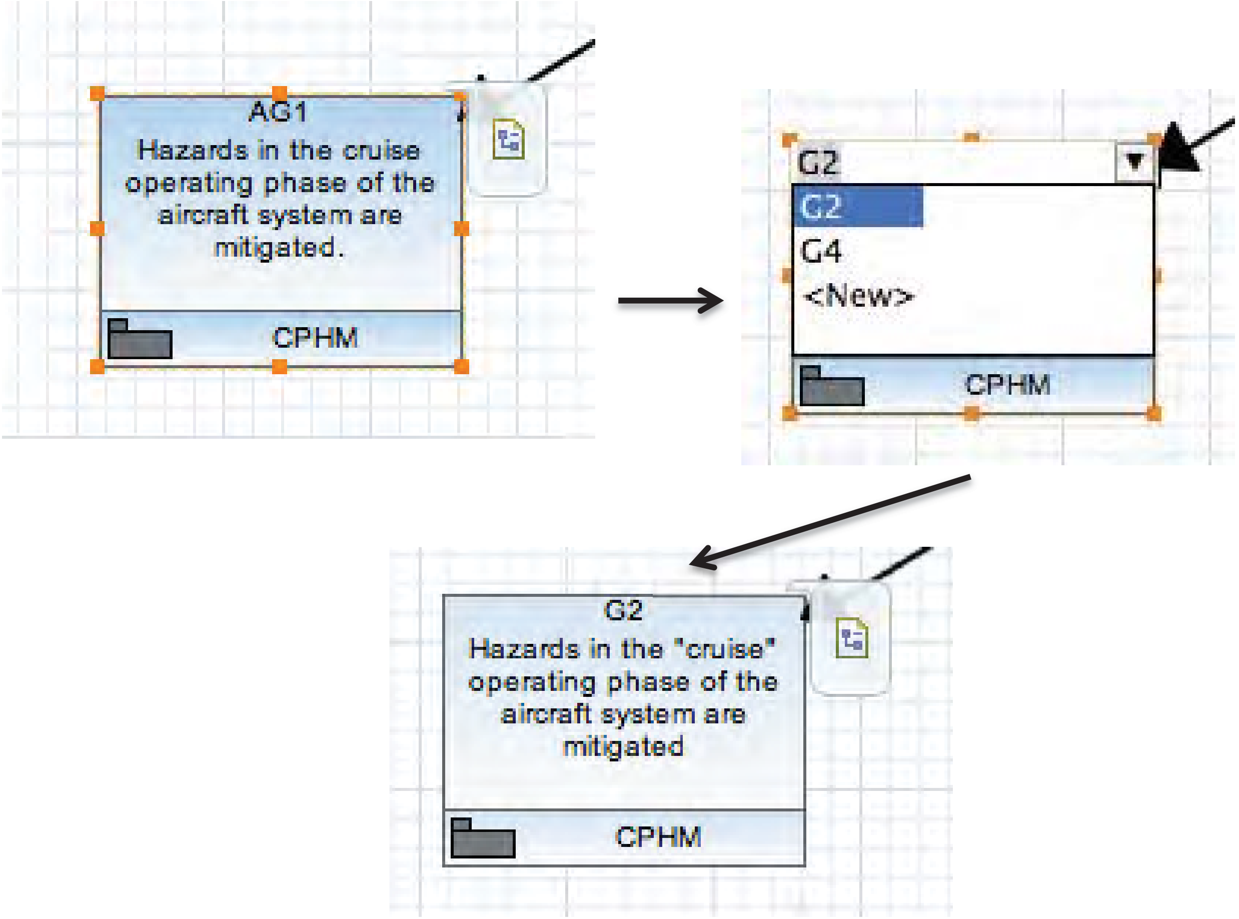
☑ Public

□ To Be Supported By Cont

Putting it All Together: Complex Diagram Walkthrough (cont.)

4. Next, the user must change the away goal node's identifier to link G1 in module CPHM by clicking the identifier and choosing the appropriate goal from the menu of public goals in CPHM.

Notice that the away goal's identifier is now G2, matching that of the goal that it is linking in CPHM. The icon at the top right of the node will link the two modules directly.

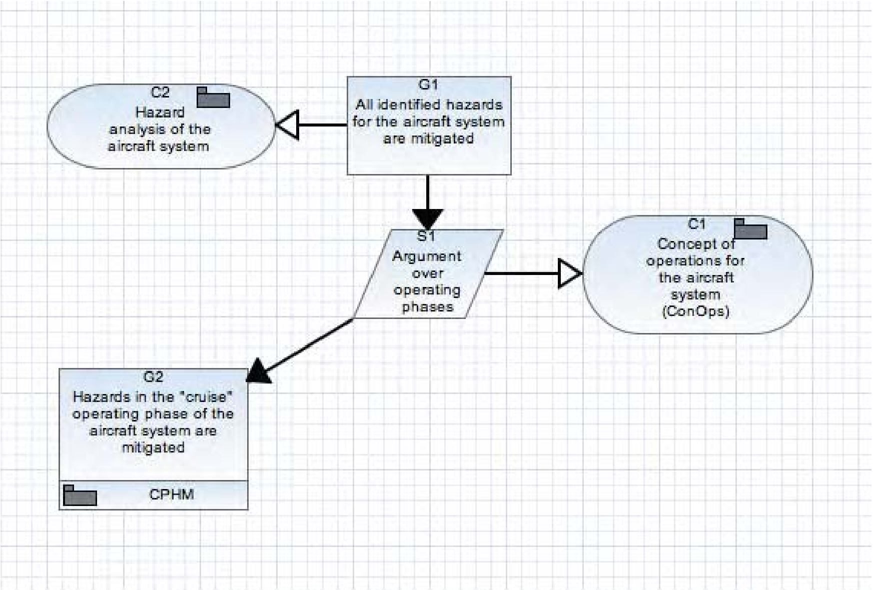


AG1
Hazards in the cruise operating phase of the aircraft system are mitigated.
CPHM

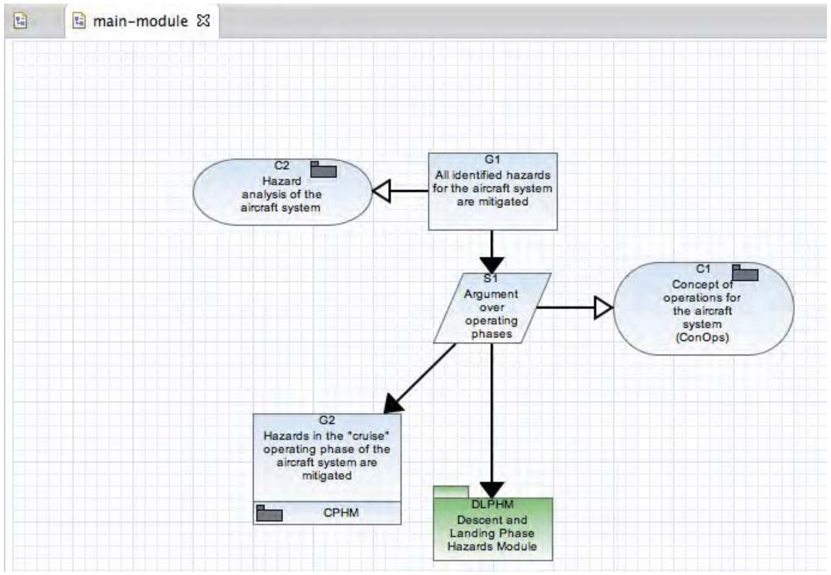
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
CPHM

Putting it All Together: Complex Diagram Walkthrough (cont.)

The main-module diagram now features an away goal linking it to CPHM. Next the user wants to add in a reference to another module DLPHM, which has not been created yet.



C2
Hazard analysis of the aircraft system
G1
All identified hazards for the aircraft system are mitigated
S1
Argument over operating phases
C1
Concept of operations for the aircraft system (ConOps)
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
CPHM

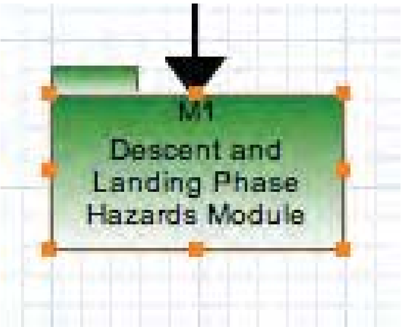


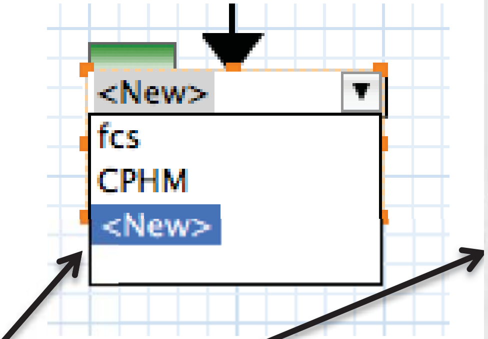
main-module
C2
Hazard analysis of the aircraft system
G1
All identified hazards for the aircraft system are mitigated
S1
Argument over operating phases
C1
Concept of operations for the aircraft system (ConOps)
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
CPHM
DLPHM
Descent and Landing Phase Hazards Module

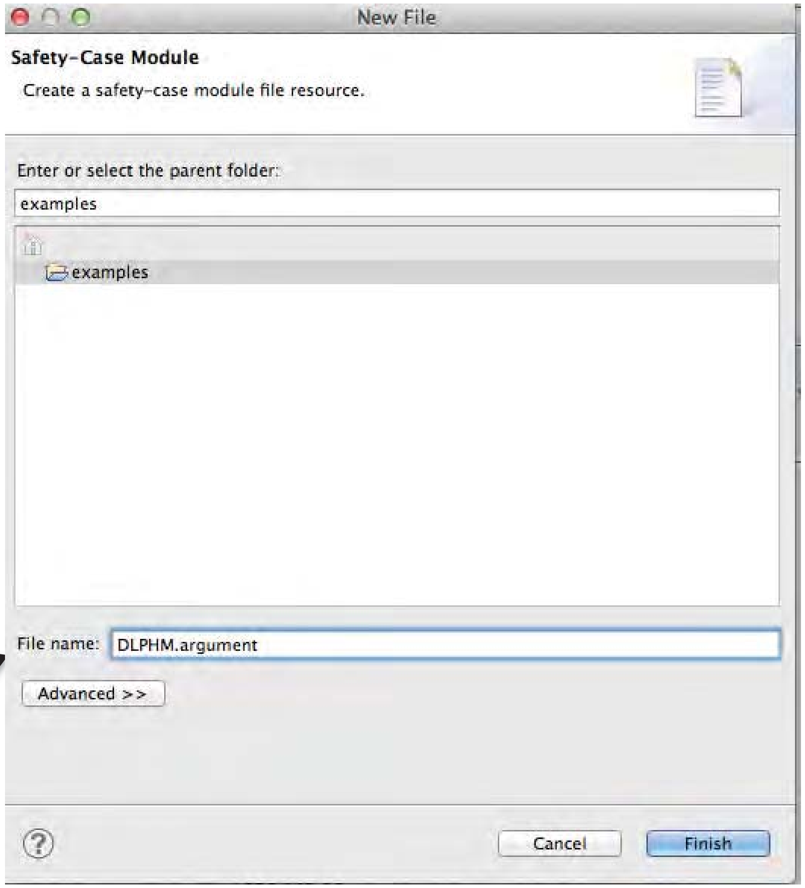
Putting it All Together: Complex Diagram Walkthrough (cont.)

The user:

1. Adds a Module Reference Node and includes the name of the module to be referenced in the node's description. In this case, the user wants to reference the Descent and Landing Phase Hazards Module (DLPHM).
2. Clicking the node's identifier will list the menu of all available modules to reference. Since DLPHM does not exist, it isn't on the list, so the user selects <New> and names a new module DLPHM. This automatically opens a new tab for that module.





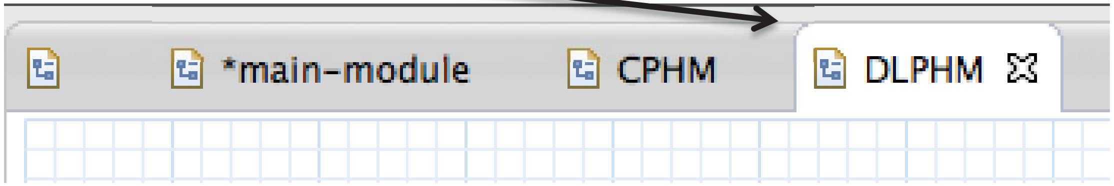

New File

Safety-Case Module

Create a safety-case module file resource.

Enter or select the parent folder: examples

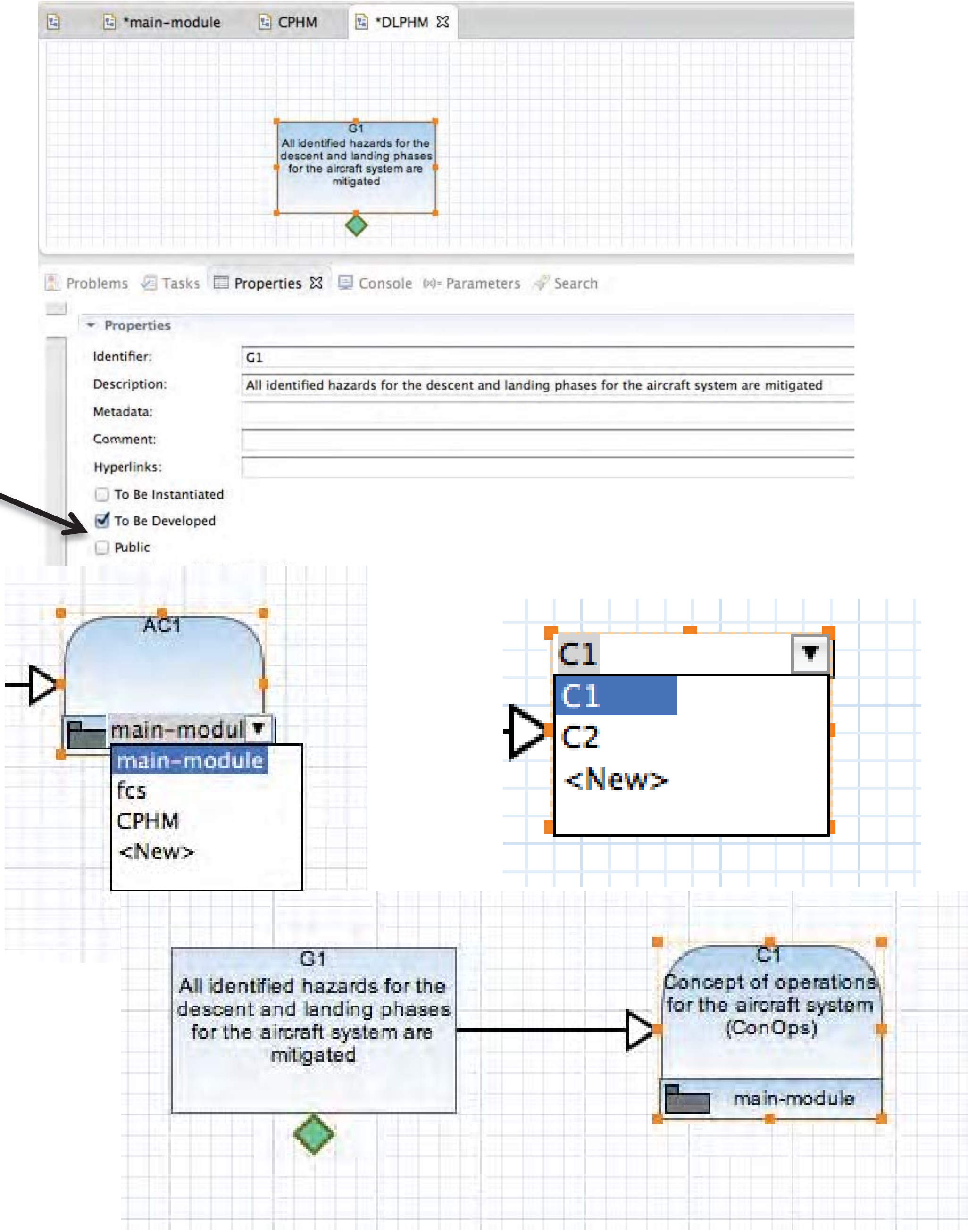
File name: DLPHM.argument



Putting it All Together: Complex Diagram Walkthrough (cont.)

The user now can create the desired DLPHM diagram. The user wants to add a goal that is under development and a context that links back to the main-module diagram. To do so, the user:

1. Adds a Goal G1 and fills in its description. In the G1 node's Properties, the user checks the box "To be developed" and a green diamond appears below the node on the diagram.
2. Adds an Away Context AC1. In the dropdown at the bottom of the node, the user selects the module to which this node will reference. In this case, it's the main-module diagram.
3. Clicks AC1 and specifies which context node in the main-module diagram to link to. In this case it's C1. The away context node's identifier and description automatically populate with the C1 node's information.



*main-module
CPHM
*DLPHM
G1
All identified hazards for the descent and landing phases for the aircraft system are mitigated
Problems
Tasks
Properties
Console
Parameters
Search
Properties

Identifier: G1

Description: All identified hazards for the descent and landing phases for the aircraft system are mitigated

Metadata:

Comment:

Hyperlinks:

□ To Be Instantiated

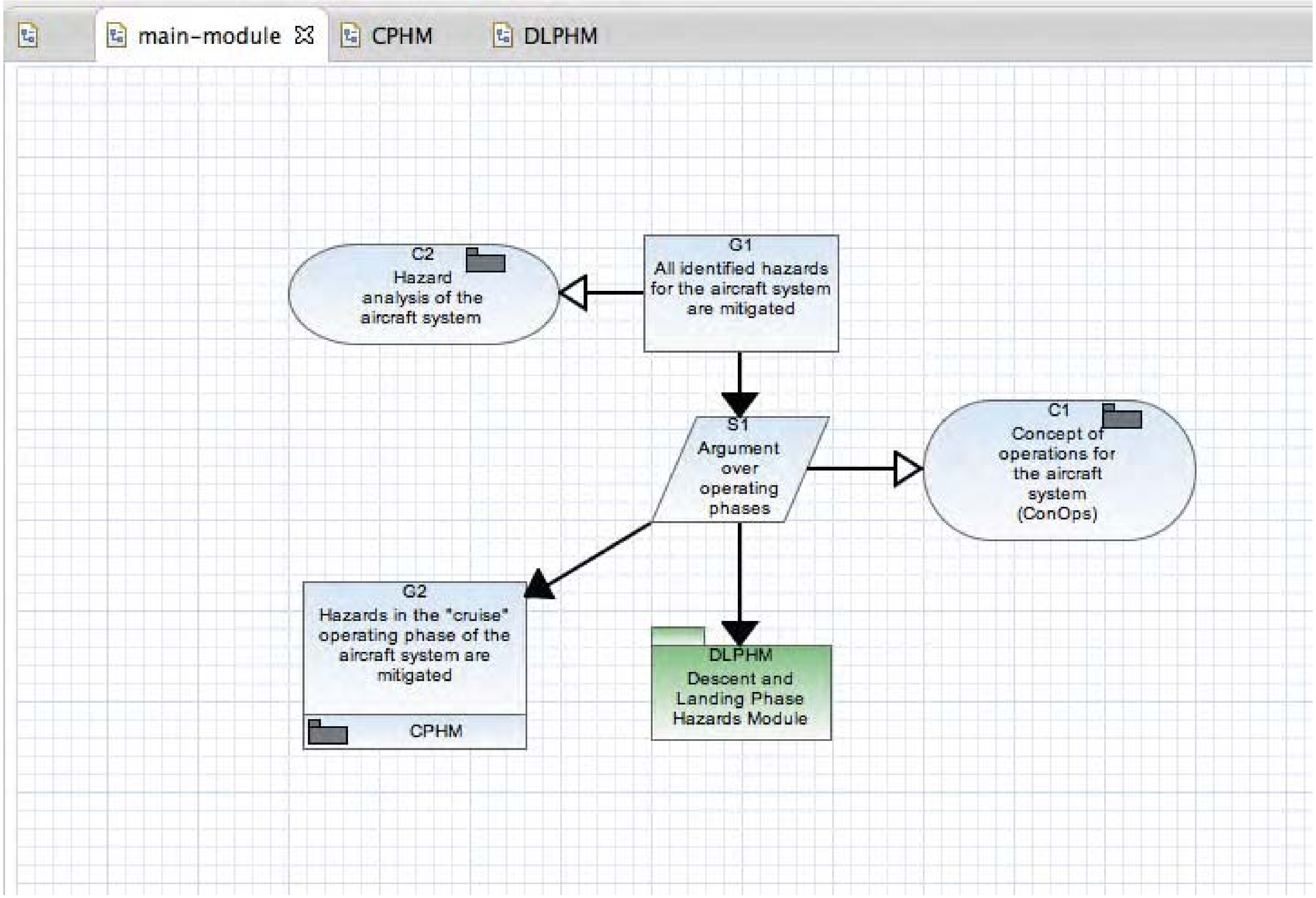
☑ To Be Developed

□ Public

AC1
main-modul
main-module
fcs
CPHM
<New>
C1
C1
C2
<New>
G1
All identified hazards for the descent and landing phases for the aircraft system are mitigated
C1
Concept of operations for the aircraft system (ConOps)
main-module

Putting it All Together: Complex Diagram Walkthrough (cont.)

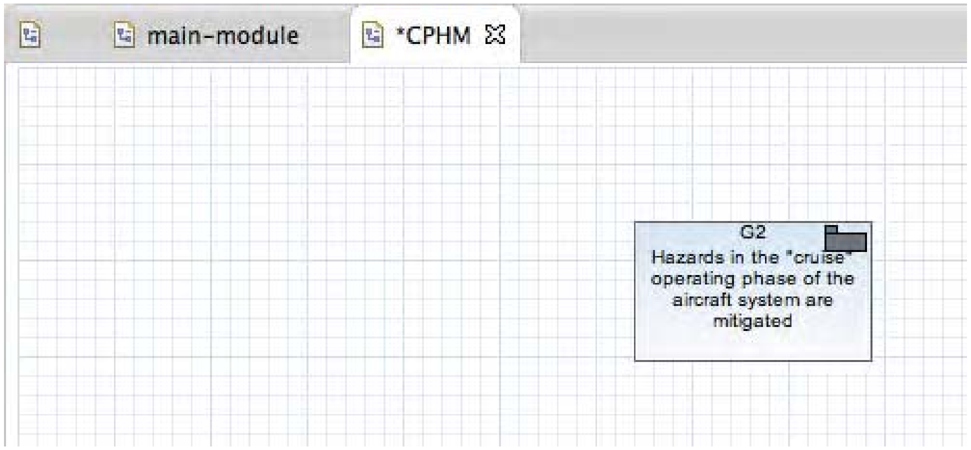
The main-module diagram now resembles the example. Next, the user wants to add some elements to the CPHM module diagram.


main-module
CPHM
DLPHM
C2
Hazard analysis of the aircraft system
G1
All identified hazards for the aircraft system are mitigated
S1
Argument over operating phases
C1
Concept of operations for the aircraft system (ConOps)
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
CPHM
DLPHM
Descent and Landing Phase Hazards Module

Putting it All Together: Complex Diagram Walkthrough (cont.)

The user wants to create a diagram branching off of G2 that includes:

- An Away Context linking back to the main-module diagram from G2
- A Strategy S1 linking from G2
- An Away Context linking back to the main-module diagram from S1
- A Goal 3 linking from S1
- A Goal 4 linking from S1
- A C-FITH contract reference to another existing module called C-FITH, which in turn references G4


main-module
*CPHM
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated

Putting it All Together: Complex Diagram Walkthrough (cont.)

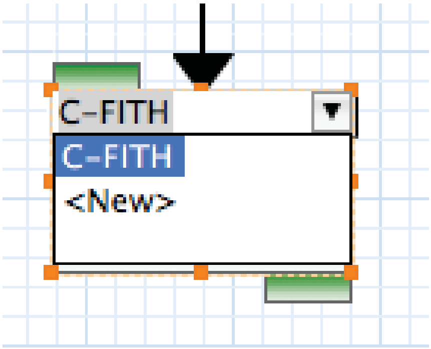
These nodes simple additions using the keyboard shortcuts.

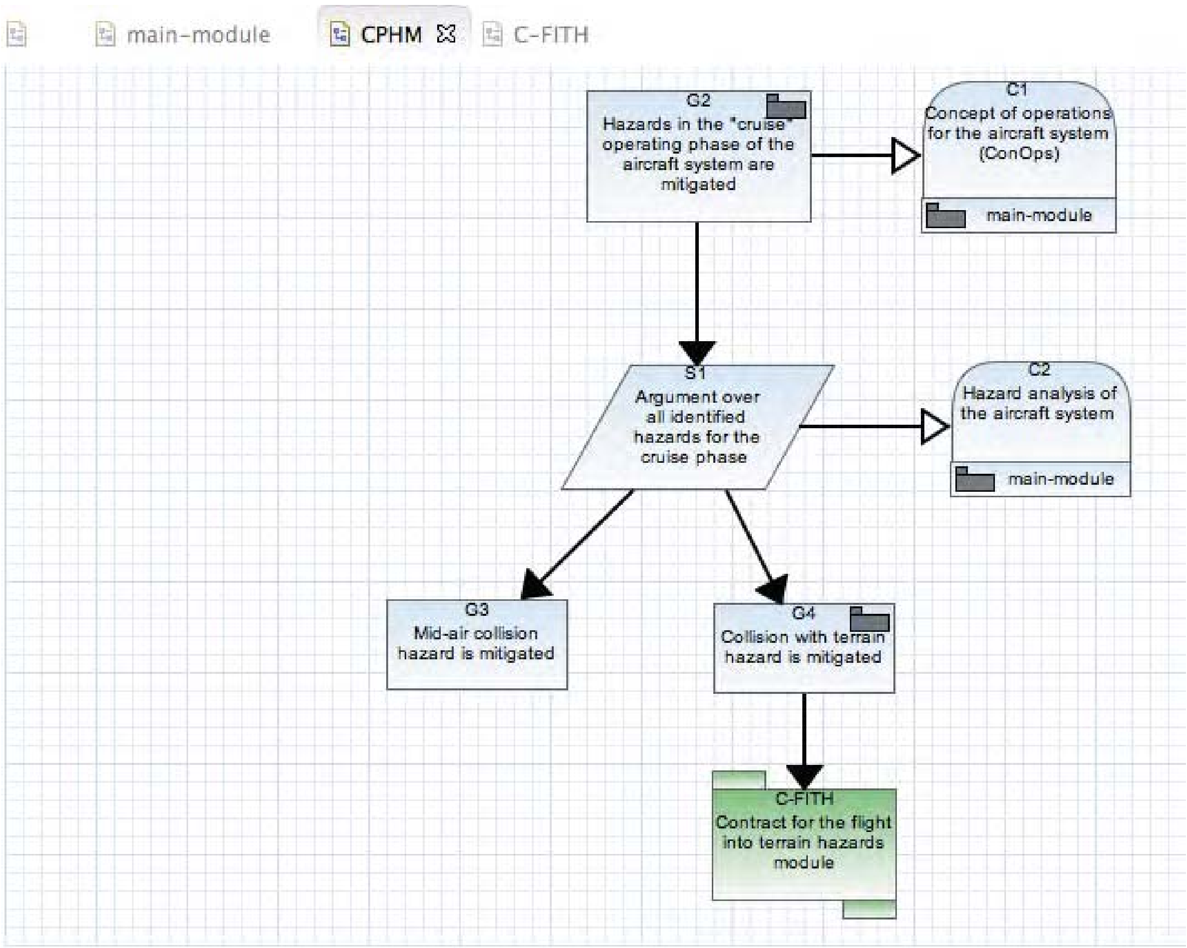
To link the away contexts to the main module, the user

1. Selects the main-module from the away context's reference menu.
2. Specifies the desired context node in the main-module to reference.

To add the contract reference node, the user:

1. Clicks the module contract's identifier and selects from the menu all of the modules

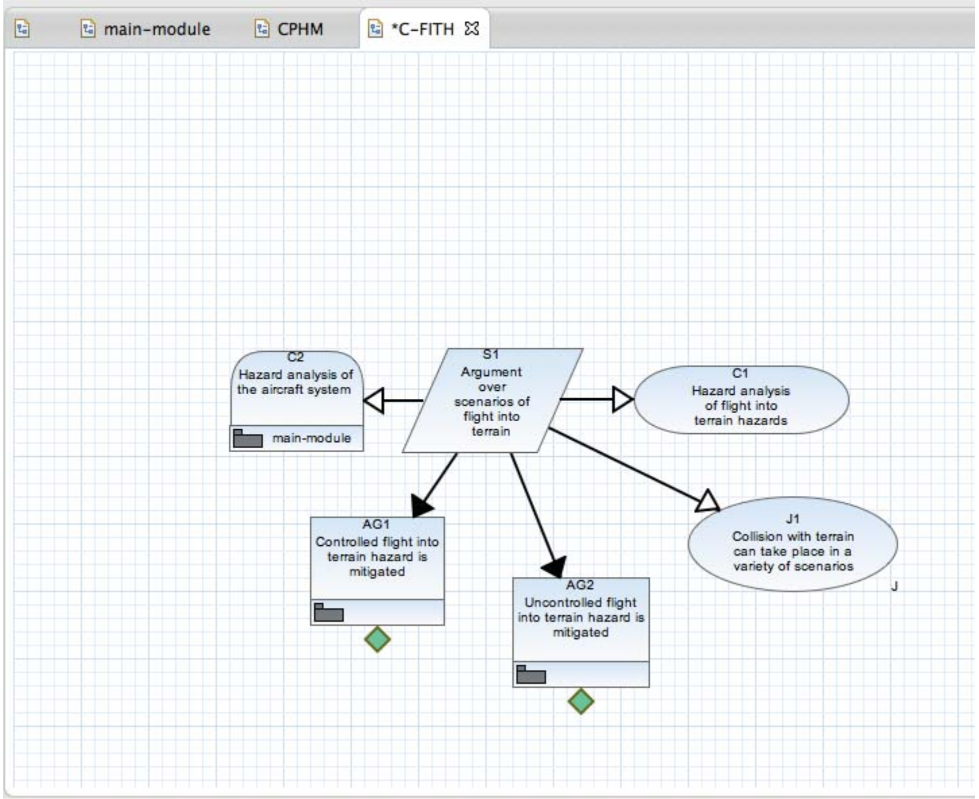



main-module
CPHM
C-FITH
G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
C1
Concept of operations for the aircraft system (ConOps)
main-module
S1
Argument over all identified hazards for the cruise phase
C2
Hazard analysis of the aircraft system
main-module
G3
Mid-air collision hazard is mitigated
G4
Collision with terrain hazard is mitigated
C-FITH
Contract for the flight into terrain hazards module

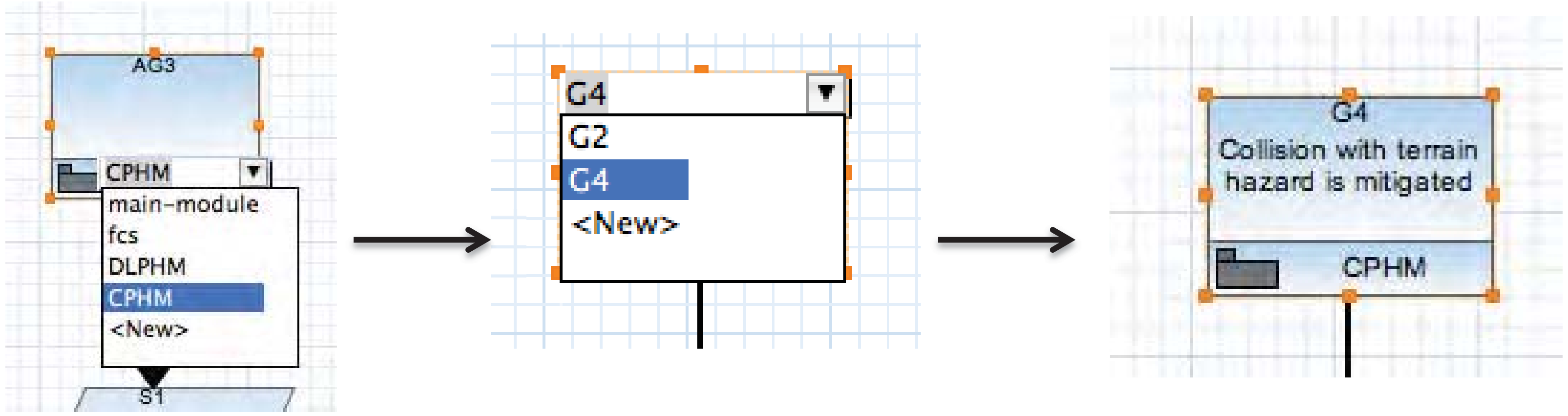
Putting it All Together: Complex Diagram Walkthrough (cont.)

The user wants to link G4 to the C-FITH module diagram via an Away Goal node. To do so, G4 must be marked public in CPHM. To create this link, the user:

1. Adds an Away Goal node to the desired location in the diagram.
2. Selects from the reference menu the module to link to. In this case, it’s CPHM.
3. Clicks the away goal node’s identifier and selects the Goal G4 node within CPHM. The identifier and description are automatically updated to match CPHM’s G4 information.



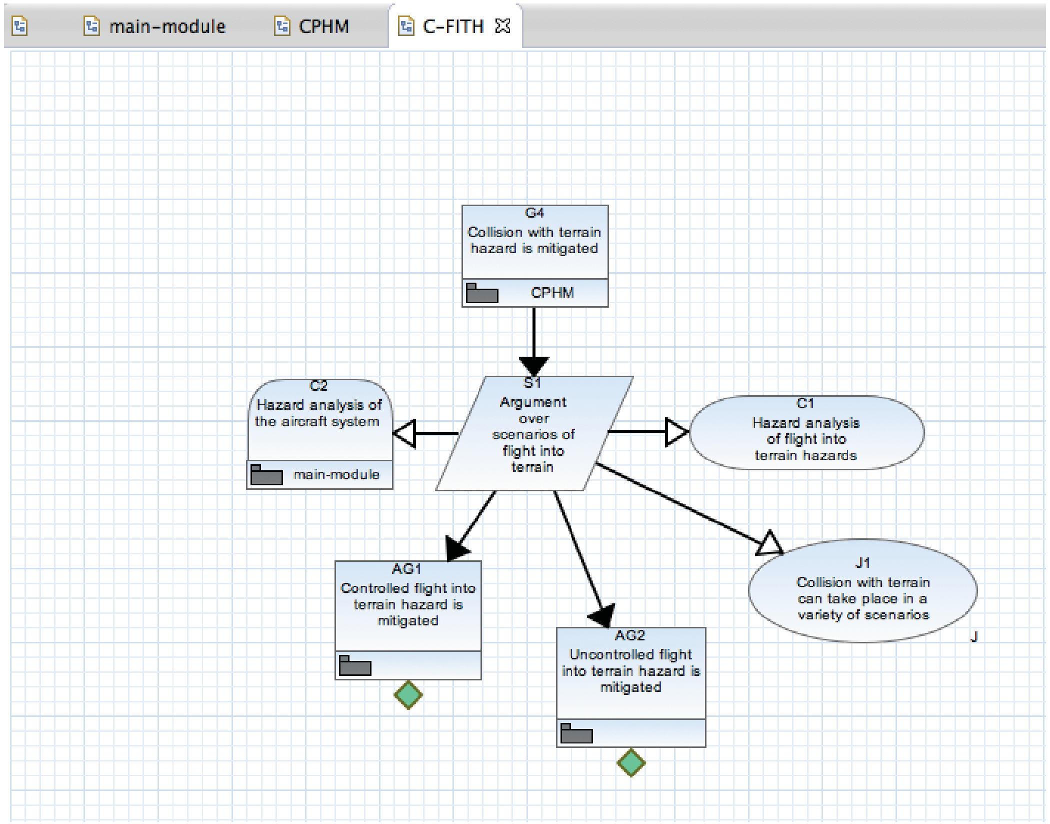
main-module
CPHM
*C-FITH
C2
Hazard analysis of the aircraft system
main-module
S1
Argument over scenarios of flight into terrain
C1
Hazard analysis of flight into terrain hazards
AG1
Controlled flight into terrain hazard is mitigated
AG2
Uncontrolled flight into terrain hazard is mitigated
J1
Collision with terrain can take place in a variety of scenarios
J



Putting it All Together: Complex Diagram Walkthrough (cont.)

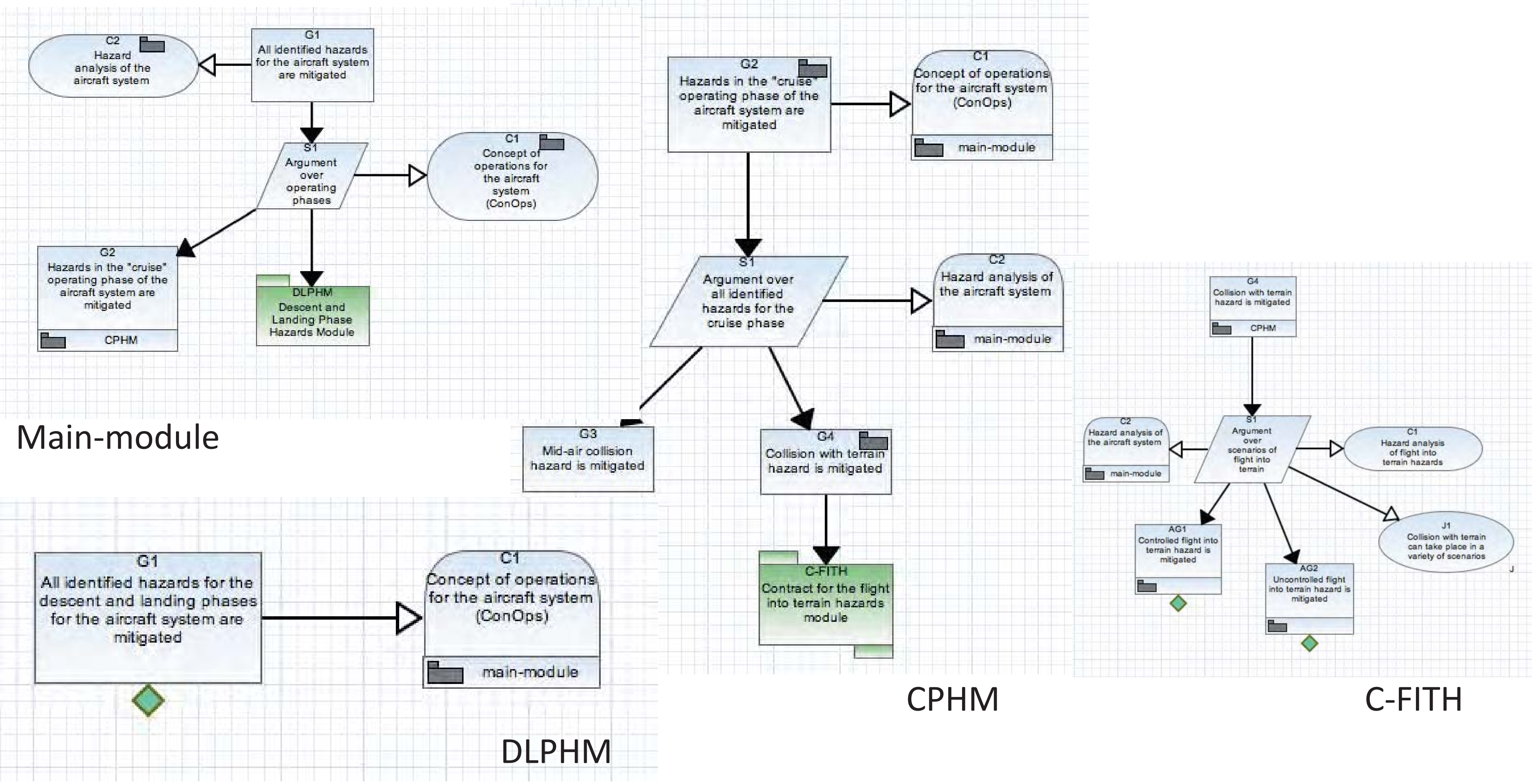
Other notes about the C-FITH module:

- Notice that the AG1 and AG2 and are not associated with any defined modules, as they are marked “To be developed.”


main-module
CPHM
C-FITH
G4
Collision with terrain hazard is mitigated
CPHM
C2
Hazard analysis of the aircraft system
main-module
S1
Argument over scenarios of flight into terrain
C1
Hazard analysis of flight into terrain hazards
J1
Collision with terrain can take place in a variety of scenarios
J
AG1
Controlled flight into terrain hazard is mitigated
AG2
Uncontrolled flight into terrain hazard is mitigated

Putting it All Together: Complex Diagram Walkthrough (cont.)

The user has created a complex diagram spanning several modules by using various links and reference tools. How a diagram is broken up is determined by the user. Some diagram elements, like the DLPHM reference node and G2 away goal in the main-module diagram, perform similar functions (linking to other modules) but can be realized in different ways.



Main-module

G1
All identified hazards for the aircraft system are mitigated

C2
Hazard analysis of the aircraft system

S1
Argument over operating phases

C1
Concept of operations for the aircraft system (ConOps)

G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated
CPHM

DLPHM
Descent and Landing Phase Hazards Module

DLPHM

G1
All identified hazards for the descent and landing phases for the aircraft system are mitigated

C1
Concept of operations for the aircraft system (ConOps)
main-module

CPHM

G2
Hazards in the "cruise" operating phase of the aircraft system are mitigated

C1
Concept of operations for the aircraft system (ConOps)
main-module

S1
Argument over all identified hazards for the cruise phase

C2
Hazard analysis of the aircraft system
main-module

G3
Mid-air collision hazard is mitigated

G4
Collision with terrain hazard is mitigated

C-FITH
Contract for the flight into terrain hazards module

C-FITH

G4
Collision with terrain hazard is mitigated
CPHM

C2
Hazard analysis of the aircraft system
main-module

S1
Argument over scenarios of flight into terrain

C1
Hazard analysis of flight into terrain hazards

AG1
Controlled flight into terrain hazard is mitigated

AG2
Uncontrolled flight into terrain hazard is mitigated

J1
Collision with terrain can take place in a variety of scenarios
J

FM Integration

Queries

Transformations

Transformations

AdvoCATE currently offers the following transformations:

1. Computing metrics: Generates a pre-determined set of metrics on the chosen argument structure

2. Creating a narrative: Generates a textual representation of the argument structure in a pre-defined format

3. CSV export: generates a comma-separated-value file in a predefined format for import into a spreadsheet to view a tabular representation of an argument structure.

4. Extracting Assumptions / Evidence / Hyperlinks: Extracts all the assumptions/evidence nodes and the specified hyperlinks in all nodes.

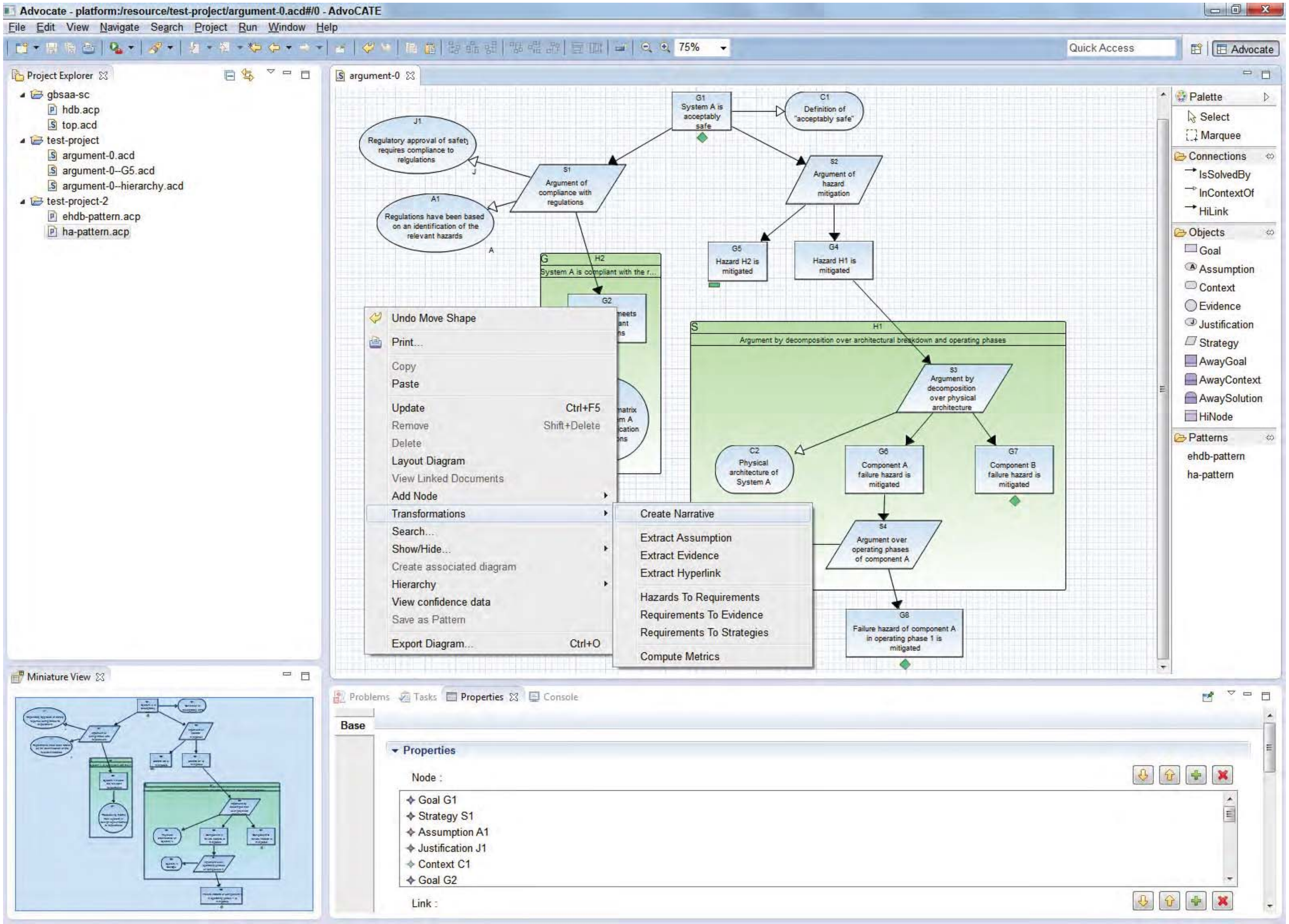
5. Traceability transformations: Lists the traceability links between Goals (with metadata marking them as *Hazards*, or *Requirements*), Strategies and Evidence. Three types of such links are generated
   a. Hazards to Requirements
   b. Requirements to Strategies
   c. Requirements to Evidence

6. Task list: Generates a list of incomplete tasks in the argument structure, i.e., nodes marked as undeveloped

⚠ Transformations can only be applied to argument structures.
They cannot be applied to argument patterns

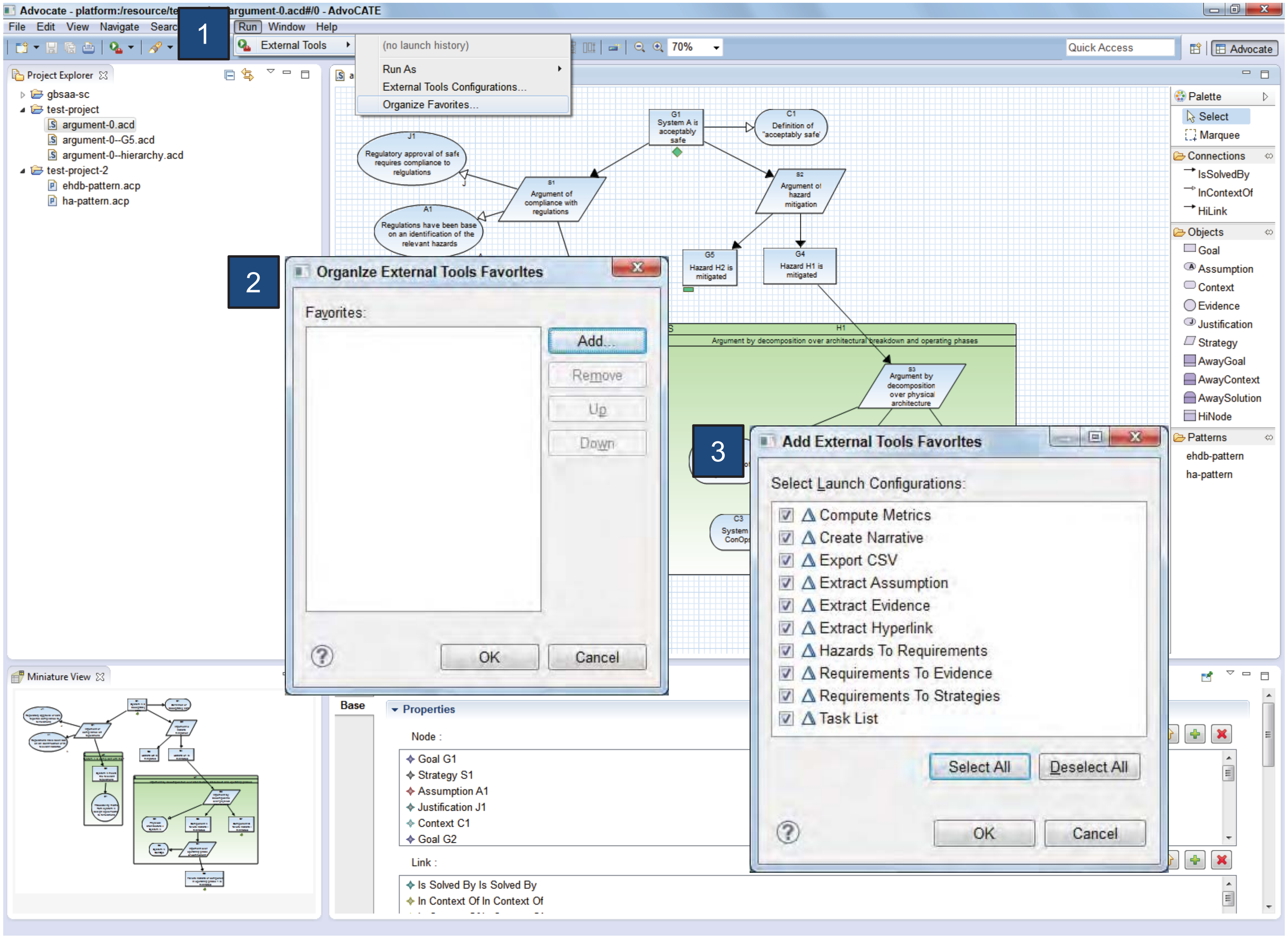
Launching Transformations

Some transformations can be launched by selecting the *Transformations* option on the right-click menu, anywhere on the canvas


Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Project Run Window Help
75%
Quick Access
Advocate
Project Explorer
gbsaa-sc
hdb.acp
top.acd
test-project
argument-0.acd
argument-0--G5.acd
argument-0--hierarchy.acd
test-project-2
ehdb-pattern.acp
ha-pattern.acp
argument-0
G1 System A is acceptably safe
C1 Definition of "acceptably safe"
J1 Regulatory approval of safety requires compliance to regulations
S1 Argument of compliance with regulations
S2 Argument of hazard mitigation
A1 Regulations have been based on an identification of the relevant hazards
H2 System A is compliant with the r...
G5 Hazard H2 is mitigated
G4 Hazard H1 is mitigated
H1 Argument by decomposition over architectural breakdown and operating phases
S3 Argument by decomposition over physical architecture
C2 Physical architecture of System A
G6 Component A failure hazard is mitigated
G7 Component B failure hazard is mitigated
S4 Argument over operating phases of component A
G8 Failure hazard of component A in operating phase 1 is mitigated
Undo Move Shape
Print...
Copy
Paste
Update Ctrl+F5
Remove Shift+Delete
Delete
Layout Diagram
View Linked Documents
Add Node
Transformations
Search...
Show/Hide...
Create associated diagram
Hierarchy
View confidence data
Save as Pattern
Export Diagram... Ctrl+O
Create Narrative
Extract Assumption
Extract Evidence
Extract Hyperlink
Hazards To Requirements
Requirements To Evidence
Requirements To Strategies
Compute Metrics
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ehdb-pattern
ha-pattern
Miniature View
Problems Tasks Properties Console
Base
Properties
Node :
Goal G1
Strategy S1
Assumption A1
Justification J1
Context C1
Goal G2
Link :

Adding Transformations to AdvoCATE

Alternatively, some or all the transformations can be added to the AdvoCATE main menu and task bar


Advocate - platform:/resource/te...argument-0.acd#0 - AdvoCATE
File Edit View Navigate Search Run Window Help
1
External Tools
(no launch history)
Run As
External Tools Configurations...
Organize Favorites...
70%
Quick Access
Advocate
Project Explorer
gbsaa-sc
test-project
argument-0.acd
argument-0--G5.acd
argument-0--hierarchy.acd
test-project-2
ehdb-pattern.acp
ha-pattern.acp
G1 System A is acceptably safe
C1 Definition of 'acceptably safe'
J1 Regulatory approval of safe requires compliance to relgulations
S1 Argument of compliance with regulations
S2 Argument of hazard mitigation
A1 Regulations have been base on an identification of the relevant hazards
G5 Hazard H2 is mitigated
G4 Hazard H1 is mitigated
H1 Argument by decomposition over architectural breakdown and operating phases
S3 Argument by decomposition over physical architecture
C3 System ConOps
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification
Strategy
AwayGoal
AwayContext
AwaySolution
HiNode
Patterns
ehdb-pattern
ha-pattern
2
Organize External Tools Favorites
Favorites:
Add...
Remove
Up
Down
OK
Cancel
3
Add External Tools Favorites
Select Launch Configurations:
Select All
Deselect All
OK
Cancel
Miniature View
Base
Properties
Node :
Link :

Select Launch Configurations:

- ☑ Compute Metrics
- ☑ Create Narrative
- ☑ Export CSV
- ☑ Extract Assumption
- ☑ Extract Evidence
- ☑ Extract Hyperlink
- ☑ Hazards To Requirements
- ☑ Requirements To Evidence
- ☑ Requirements To Strategies
- ☑ Task List

Node :

- Goal G1
- Strategy S1
- Assumption A1
- Justification J1
- Context C1
- Goal G2

Link :

- Is Solved By Is Solved By
- In Context Of In Context Of

To add transformations

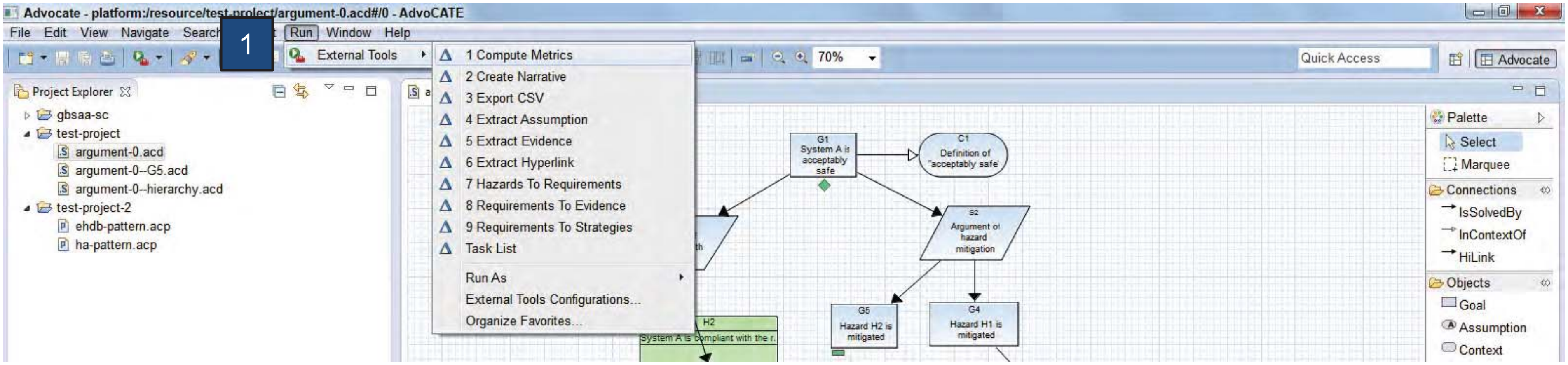
1. select from the main menu, the option *Run > External Tools > Organize Favorites*
2. select *Add* from the dialog box that opens
3. Then select all (or the desired) transformations from the list.

Launching Transformations

After adding transformations, launch them in one of two ways:

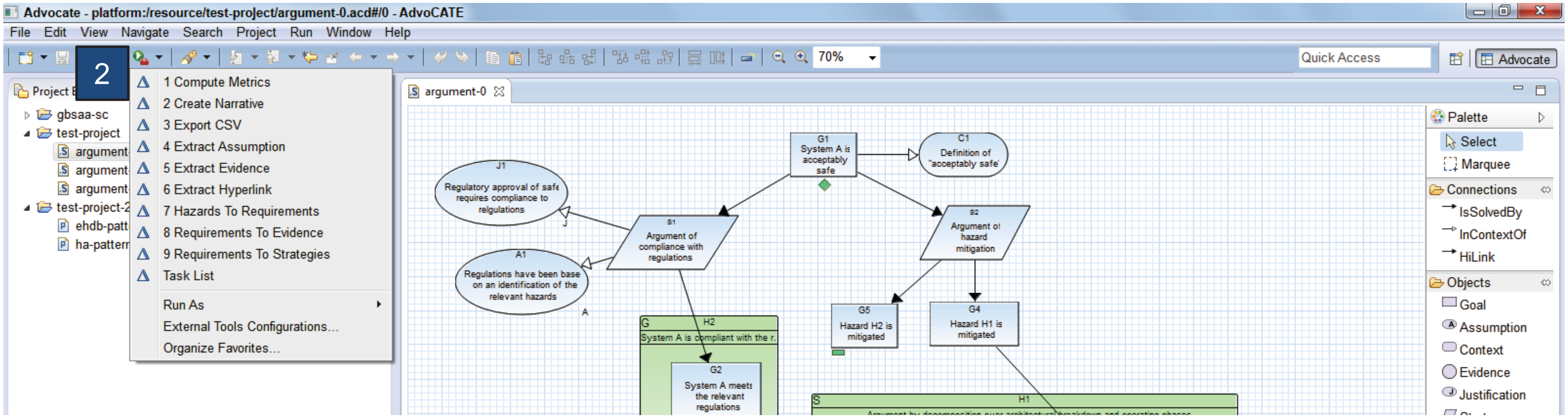
First, select the resource in the project explorer. Then,

1. select from the main menu the option *Run* > *External Tools* , then select the desired transformation.



Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Run Window Help
1
External Tools
1 Compute Metrics
2 Create Narrative
3 Export CSV
4 Extract Assumption
5 Extract Evidence
6 Extract Hyperlink
7 Hazards To Requirements
8 Requirements To Evidence
9 Requirements To Strategies
Task List
Run As
External Tools Configurations...
Organize Favorites...
70%
Quick Access
Advocate
Project Explorer
gbsaa-sc
test-project
argument-0.acd
argument-0--G5.acd
argument-0--hierarchy.acd
test-project-2
ehdb-pattern.acp
ha-pattern.acp
G1 System A is acceptably safe
C1 Definition of "acceptably safe"
S2 Argument of hazard mitigation
G5 Hazard H2 is mitigated
G4 Hazard H1 is mitigated
H2 System A is compliant with the r...
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context

2. Alternatively, run the desired transformation from the *Run* shortcut on the task bar



Advocate - platform:/resource/test-project/argument-0.acd#/0 - AdvoCATE
File Edit View Navigate Search Project Run Window Help
2
1 Compute Metrics
2 Create Narrative
3 Export CSV
4 Extract Assumption
5 Extract Evidence
6 Extract Hyperlink
7 Hazards To Requirements
8 Requirements To Evidence
9 Requirements To Strategies
Task List
Run As
External Tools Configurations...
Organize Favorites...
70%
Quick Access
Advocate
Project E
gbsaa-sc
test-project
argument
argument
argument
test-project-2
ehdb-patt
ha-patter
argument-0
J1 Regulatory approval of safe requires compliance to relgulations
A1 Regulations have been base on an identification of the relevant hazards
S1 Argument of compliance with regulations
G1 System A is acceptably safe
C1 Definition of "acceptably safe"
S2 Argument of hazard mitigation
G5 Hazard H2 is mitigated
G4 Hazard H1 is mitigated
H2 System A is compliant with the r...
G2 System A meets the relevant regulations
H1
Palette
Select
Marquee
Connections
IsSolvedBy
InContextOf
HiLink
Objects
Goal
Assumption
Context
Evidence
Justification

Configuring Transformations

Transformations can be configured, so that their output either appears in the *Console* panel, is written to a specified folder on disk, or both.

Import and Export

Importing Other Formats

AdvoCATE can import arguments in the D-CASE and ASCE formats.

Merging Formal Arguments

AdvoCATE can create arguments from the output of formal methods tools (currently just AutoCert).

⚠ This feature will change significantly in future. It will be systematized to include other tools.

Troubleshooting

Troubleshooting

- Default workspaces
- What if AdvoCATE opens with all of the panels minimized?
- Workspace corrupted—create projects external from workspace, re-import projects
- If you can’t move the nodes—collisions