



ASCE v4 display engine

Luke Emmet

loe@adelard.com

5th November 2008

Overview

- ASCE v3 display engine
- Requirements and use cases
- ASCE v4 display engine

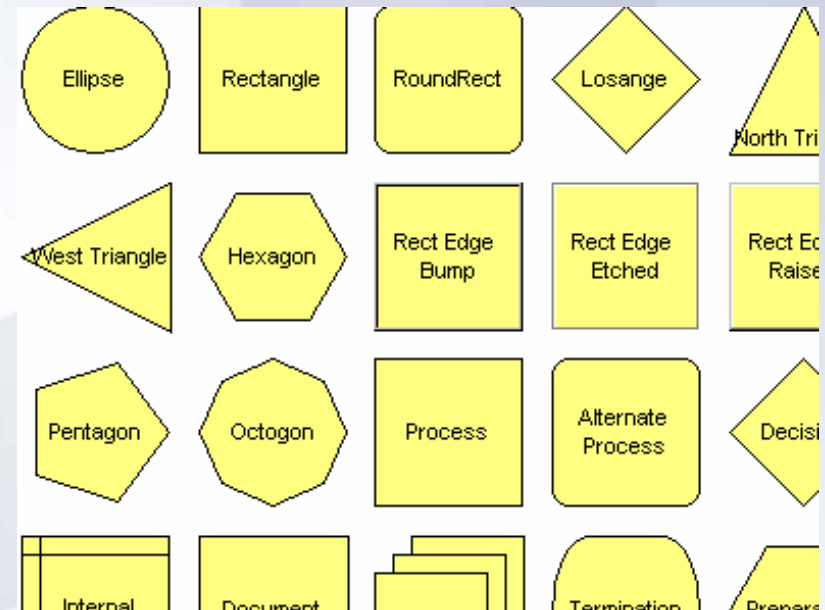
- demo

ASCE v3 engine

- Based around schema definition:
 - Node types
 - Status fields (node attributes)
 - Link Types
 - Display rules
 - Check rules

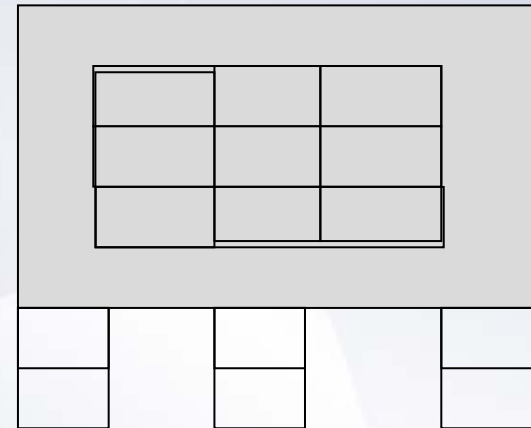
ASCE v3 - node types

- Visually built as a static layering of shapes from a fixed palette
- Draw and fill colour
 - Blue ellipse is Claim in CAE
- Text of shape components
 - (e.g. J of "Justification" in GSN)
- Display text at some node is the node id and title



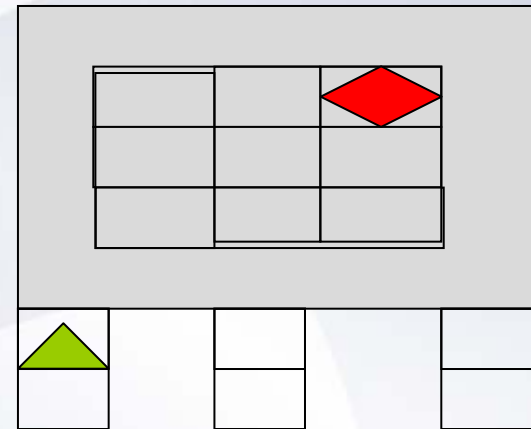
ASCE v3 display rules

- Display rules provide decorators on top of the basic node shape
- A number of predefined positions
 1. Background of node itself
 2. 3x3 grid
 3. "hanging positions" on bottom of node



ASCE v3 display rules

- The schema display rules can “light up” these positions:
 - with a specified shape and colour
 - based on a query on the node’s properties
- This gives a visual cue to a changed property
- This is how traffic lights and GSN decorators are implemented



Limitations

- Display engine has served very well so far
 - About 30 or so custom schemas have been developed
 - including the old favourites CAE and GSN
 - Introduction of “traffic lights” most popular extension
- But is limited in a number of ways
- Display text at some node limited to node id and title only
 - Displaying other properties has to be a hack
 - (e.g. Fault tree macro copies probability to id field)
 - Need a better way to do this
- Decorators are fairly limited
- Node shapes are limited to fixed palette

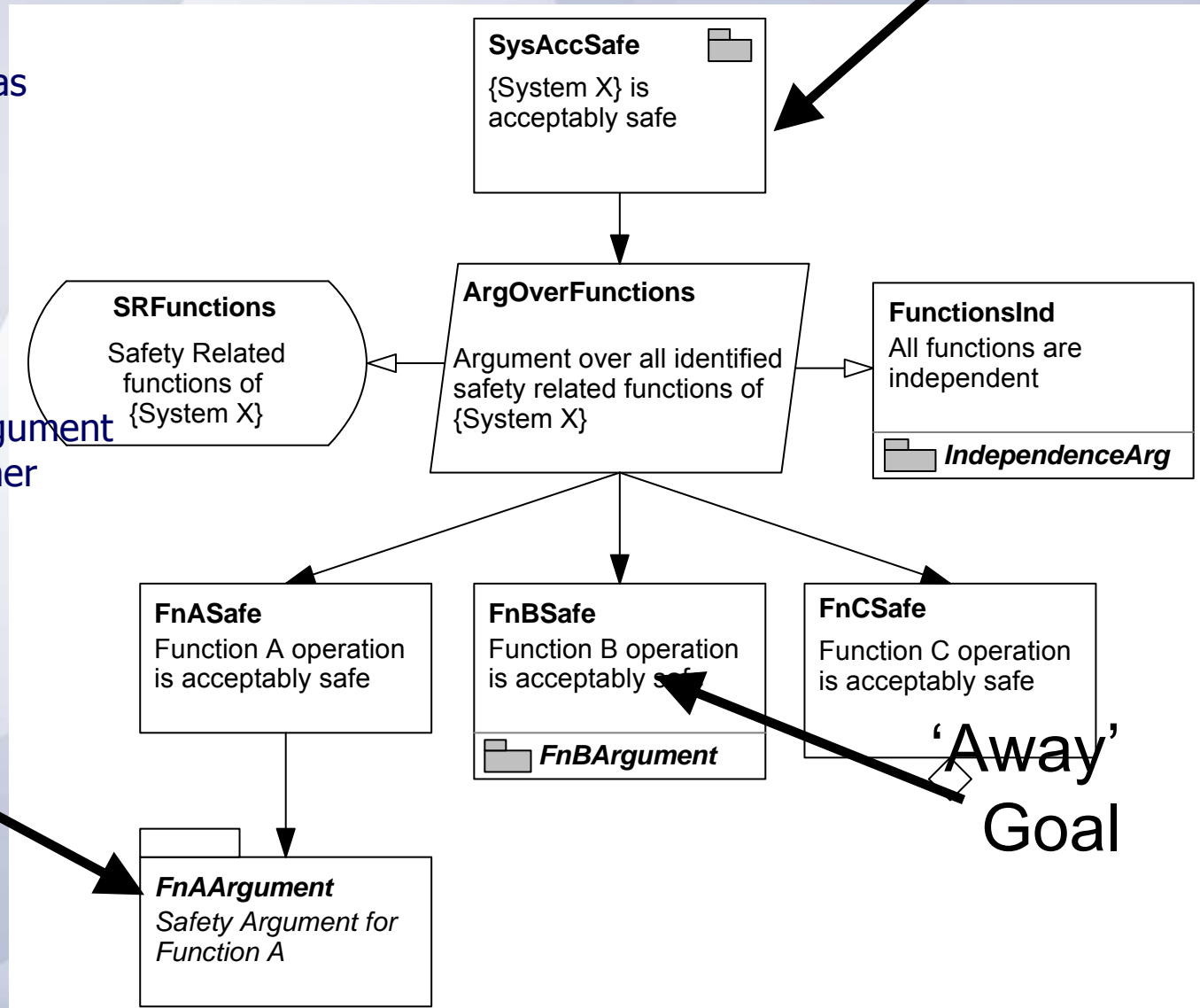
Requirements

- For v4 we want to extend display engine as a key requirement
- To support wider range of applications
 - GSN “modular extensions”
 - Better fault trees
 - ECF (events and causal factor charts)
- Innovative applications we might create
- Innovative applications you might create

Modular GSN extensions

Public Goal

- Extensions:
- Ability to mark a goal as 'public'
- Ability to refer to goals defined in other modules
- Ability to refer to modules
- Ability to place one argument in the context of another



Module Reference

'Away' Goal

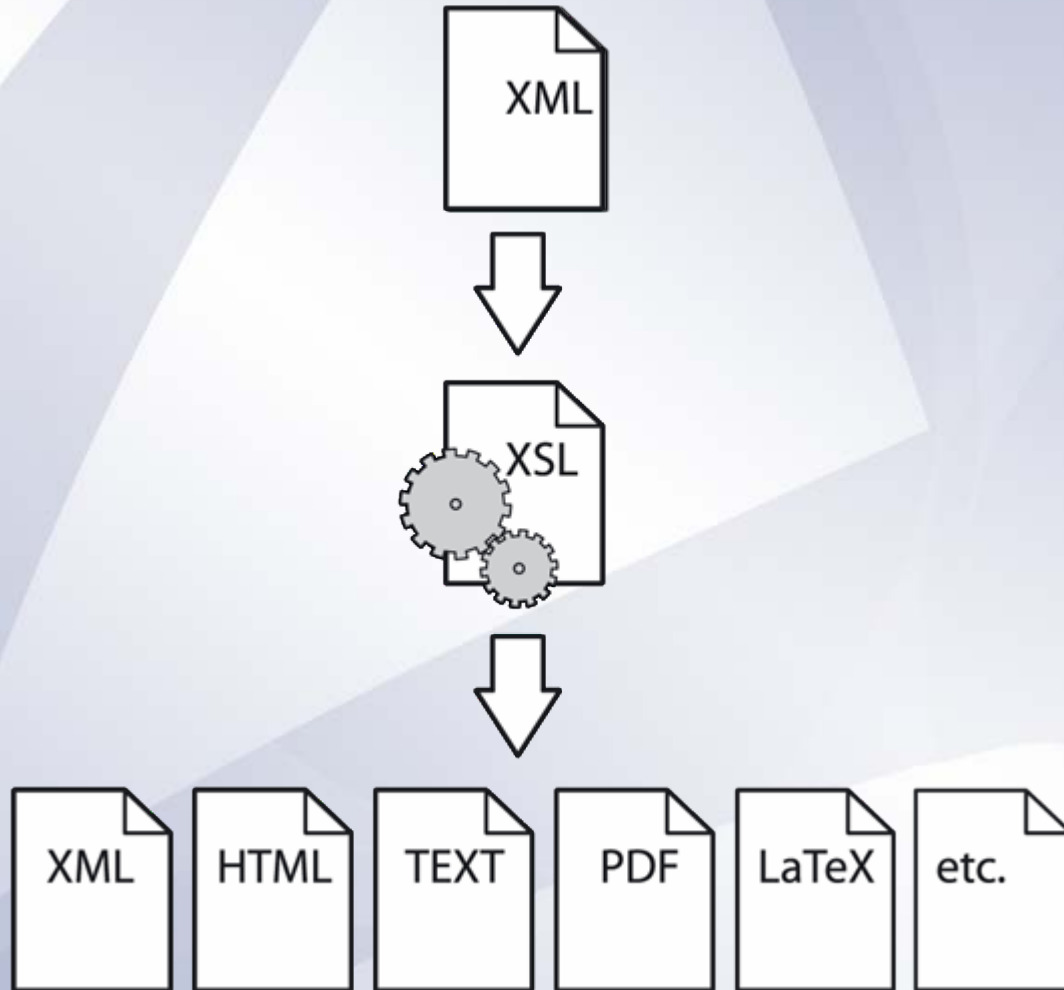
Display requirements

- Need a more flexible display engine
 - Arbitrary shapes
 - Better decorators
 - Being able to display more node properties on the node
- Work better with the plugin engine
 - Clicking on “hyperlink” on a node starts a plugin action
 - launches an external file
 - Custom editor managed by a plugin
 - Contextual feedback or analysis
 - ...

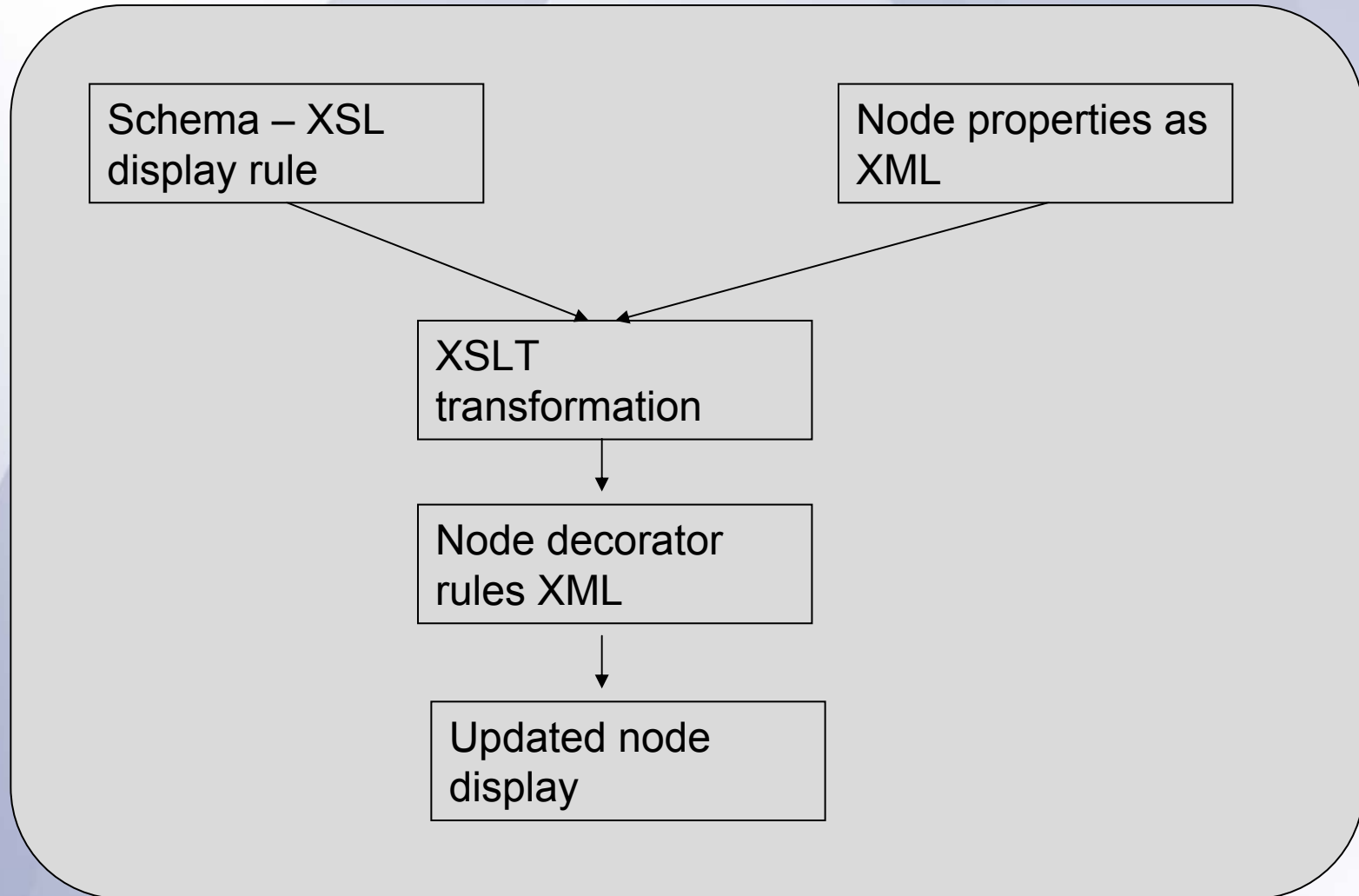
Design so far

- ASCE v4 display engine is being actively developed at the moment
 - We have design approach and partial implementation
- Uses XSL in schema to allow arbitrary node display construction based on “decorators”
 - Decorators are layered,
 - Can be anywhere on the node
 - Can display node property values as text
 - Can use predefined polygons (part of schema)

How XSL works



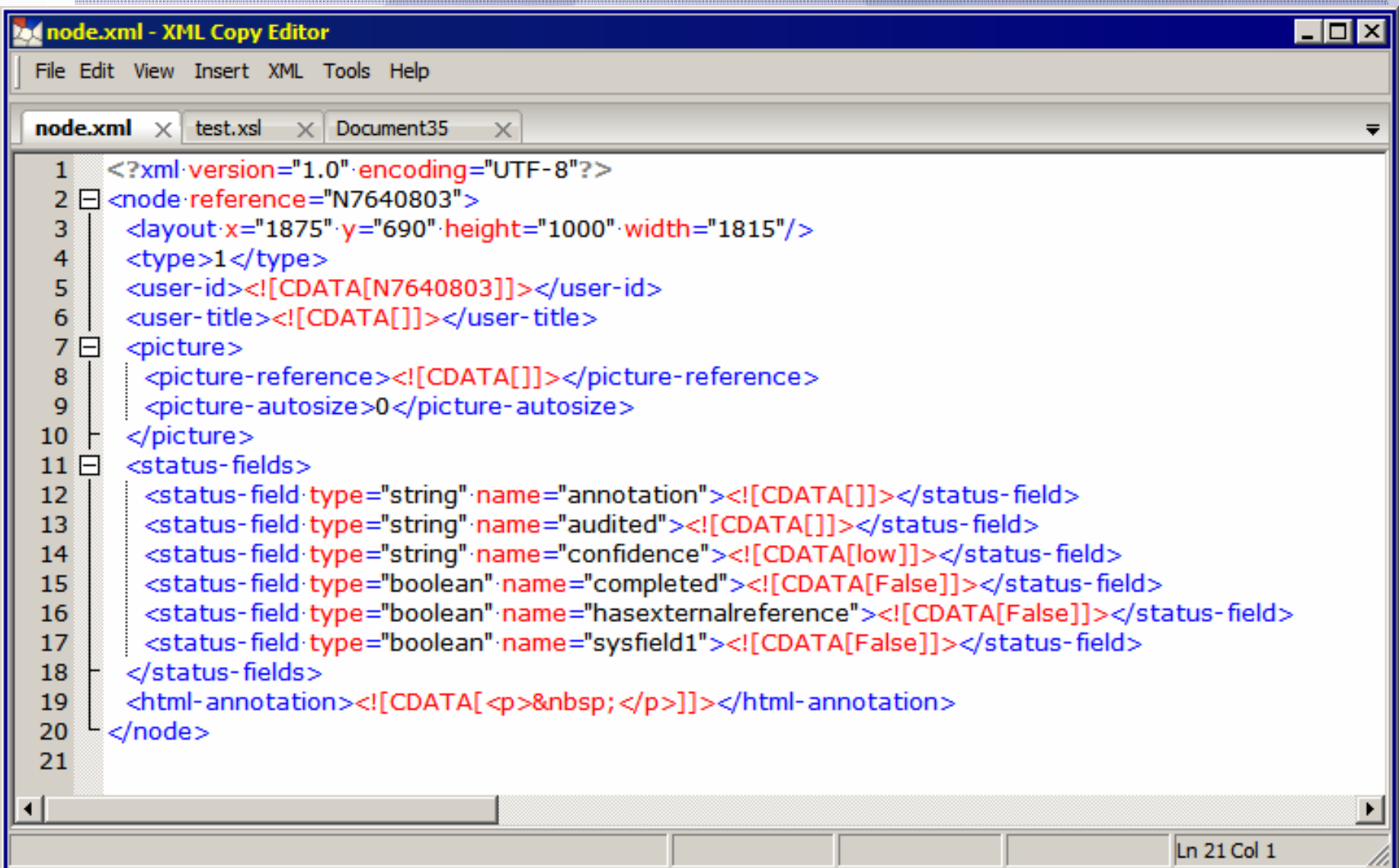
XSL display engine



Node XML fragment (similar to AXML file format)

```
<node reference="N7640803">
  <layout x="1875" y="690" height="1000" width="1815"/>
  <type>1</type>
  <user-id>N7640803</user-id>
  <user-title></user-title>
  <status-fields>
    <status-field type="string" name="annotation"></status-field>
    <status-field type="string" name="audited"></status-field>
    <status-field type="string" name="confidence">low</status-field>
    <status-field type="boolean"
name="completed">False</status-field>
  </status-fields>
</node>
```

Node content



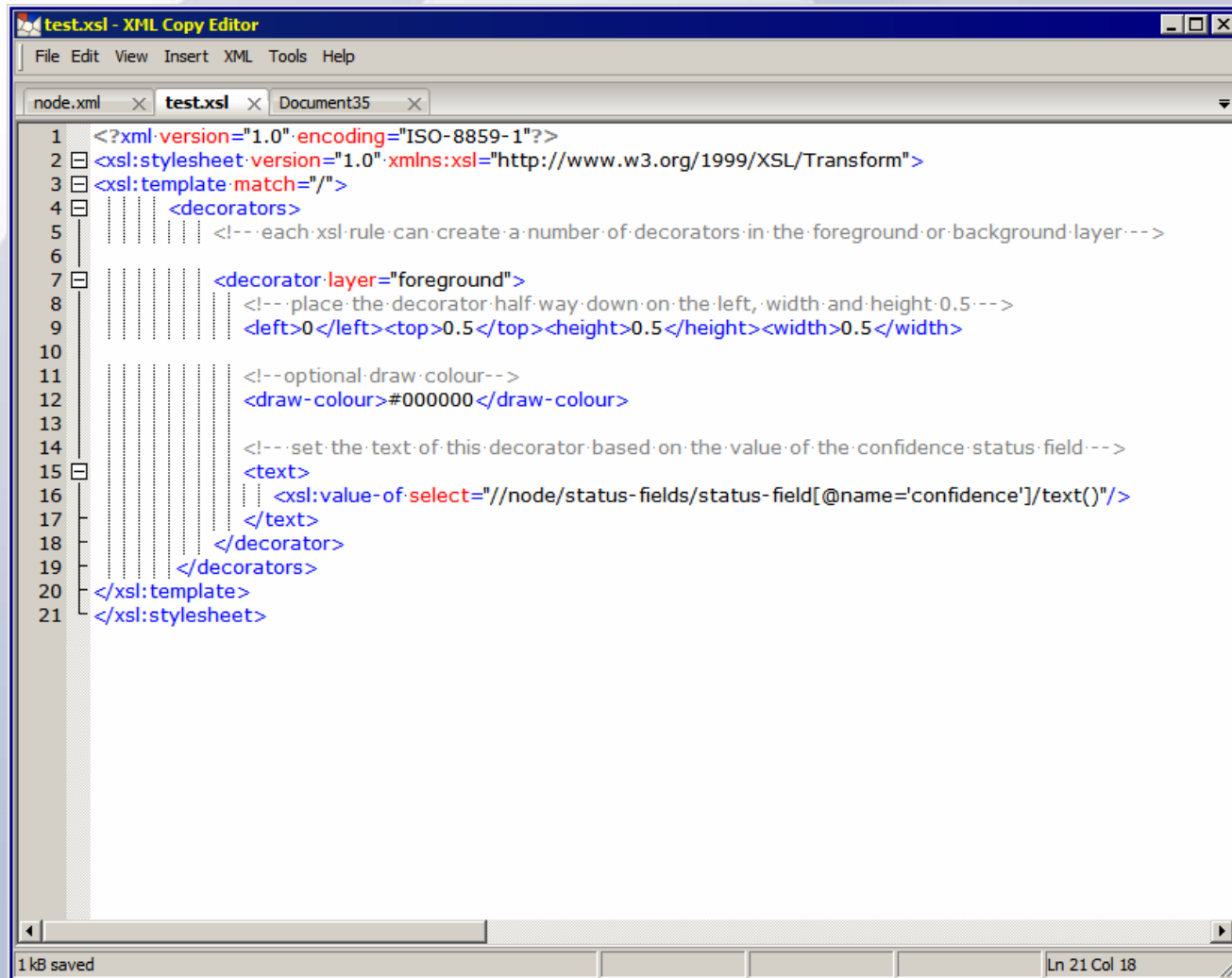
The screenshot shows a window titled "node.xml - XML Copy Editor" with a menu bar (File, Edit, View, Insert, XML, Tools, Help) and a tab bar containing "node.xml", "test.xml", and "Document35". The main text area displays XML code with line numbers 1 through 21. The code defines an XML node with various attributes and child elements. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <node reference="N7640803">
3   <layout x="1875" y="690" height="1000" width="1815"/>
4   <type>1</type>
5   <user-id><![CDATA[N7640803]]></user-id>
6   <user-title><![CDATA[]]></user-title>
7   <picture>
8     <picture-reference><![CDATA[]]></picture-reference>
9     <picture-autosize>0</picture-autosize>
10  </picture>
11  <status-fields>
12    <status-field type="string" name="annotation"><![CDATA[]]></status-field>
13    <status-field type="string" name="audited"><![CDATA[]]></status-field>
14    <status-field type="string" name="confidence"><![CDATA[low]]></status-field>
15    <status-field type="boolean" name="completed"><![CDATA[False]]></status-field>
16    <status-field type="boolean" name="hasexternalreference"><![CDATA[False]]></status-field>
17    <status-field type="boolean" name="sysfield1"><![CDATA[False]]></status-field>
18  </status-fields>
19  <html-annotation><![CDATA[<p>&nbsp;</p>]]></html-annotation>
20 </node>
21
```

The status bar at the bottom right indicates "Ln 21 Col 1".

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<decorators>
<decorator layer="foreground">
<left>0</left> <top>0.5</top> <height>0.5</height> <width>0.
  5</width>
<draw-colour> #000000</draw-colour>
<text> <xsl:value-of select="//node/status-fields/status-
  field[@name='confidence']/text()"/> </text>
</decorator>
</decorators>
</xsl:template>
</xsl:stylesheet>
```

Display rule - XSL



```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3 <xsl:template match="/">
4 <decorators>
5     <!-- each xsl rule can create a number of decorators in the foreground or background layer -->
6
7     <decorator layer="foreground">
8         <!-- place the decorator half way down on the left, width and height 0.5 -->
9         <left>0</left><top>0.5</top><height>0.5</height><width>0.5</width>
10
11         <!-- optional draw colour -->
12         <draw-colour>#000000</draw-colour>
13
14         <!-- set the text of this decorator based on the value of the confidence status field -->
15         <text>
16             <xsl:value-of select="//node/status-fields/status-field[@name='confidence']/text()"/>
17         </text>
18     </decorator>
19 </decorators>
20 </xsl:template>
21 </xsl:stylesheet>
```

1 kB saved | Ln 21 Col 18

Derived decorator

```
<decorators>  
  <decorator layer="foreground">  
    <left>0</left>  
    <top>0.5</top>  
    <height>0.5</height>  
    <width>0.5</width>  
    <draw-colour>#000000</draw-colour>  
    <text>low</text>  
  </decorator>  
</decorators>
```

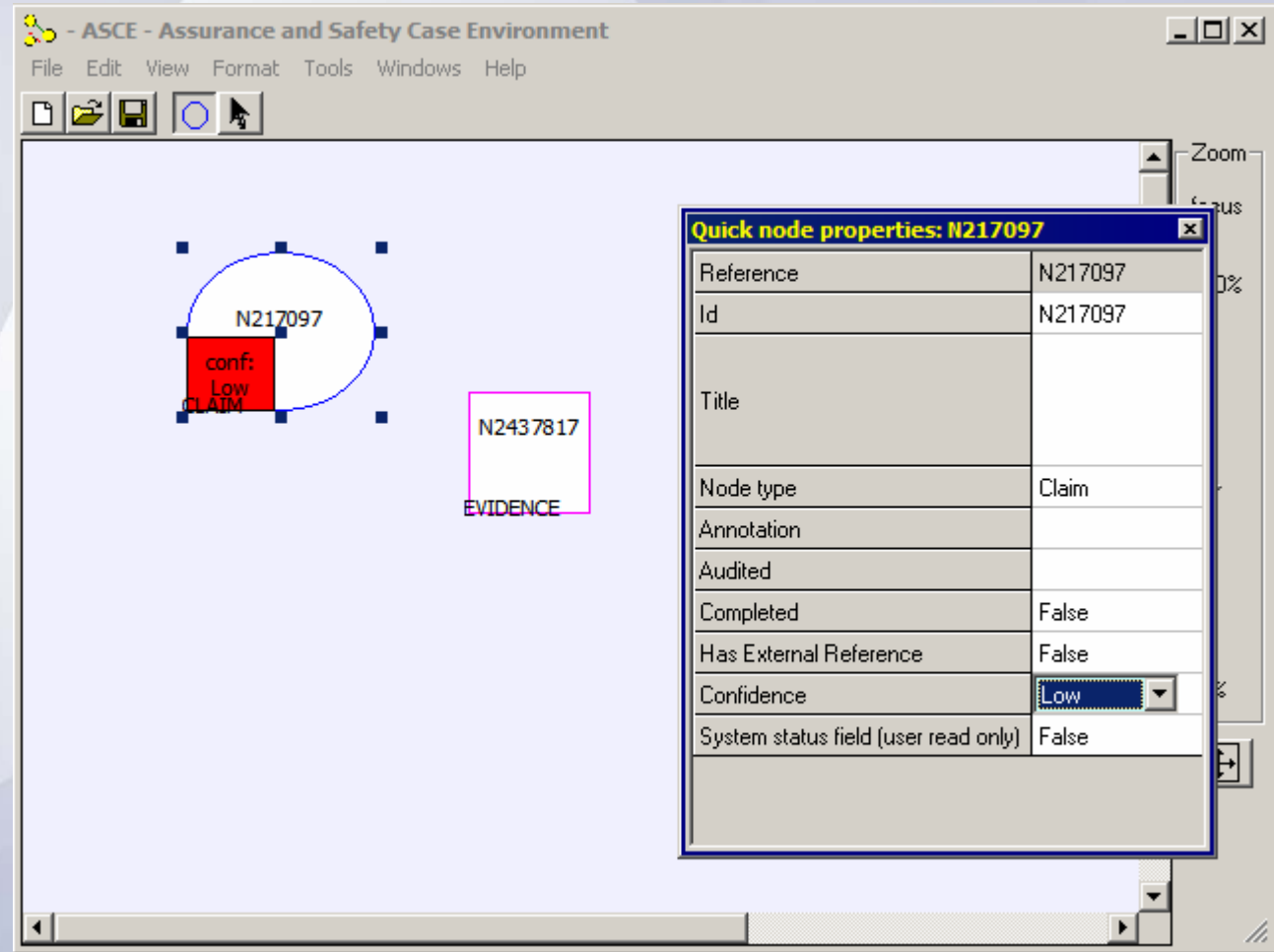
Display engine

The screenshot shows the ASCE Assurance and Safety Case Environment software. The main workspace contains a diagram with two nodes: a circular node labeled 'N217097' with the word 'CLAIM' below it, and a rectangular node labeled 'N2437817' with the word 'EVIDENCE' below it. A 'Quick node properties: N217097' dialog box is open, displaying the following table:

Quick node properties: N217097	
Reference	N217097
Id	N217097
Title	
Node type	Claim
Annotation	
Audited	
Completed	False
Has External Reference	False
Confidence	Off
System status field (user read only)	False

Display engine

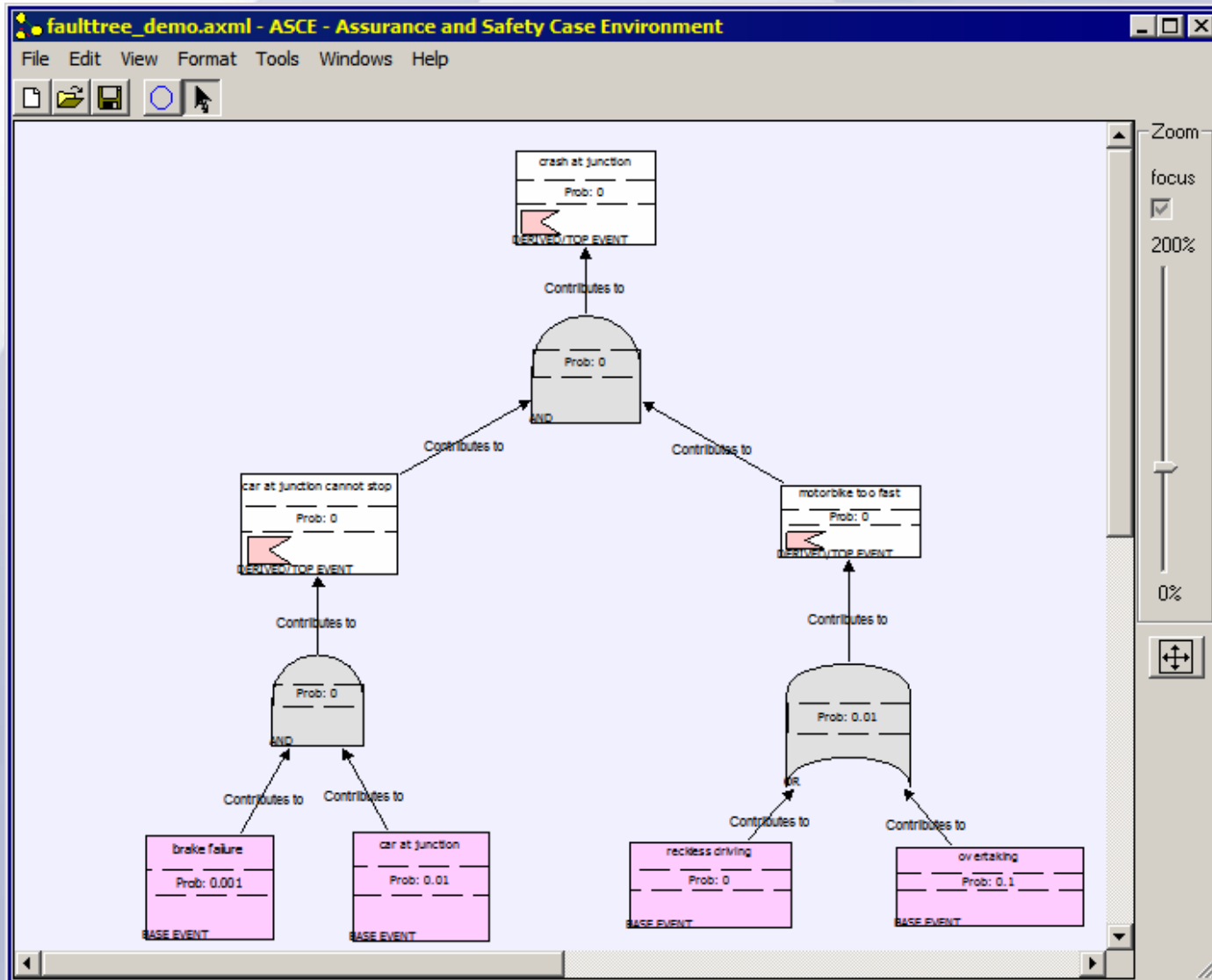
- Screenshot shows traffic light style decorator
- Displays value of confidence field on node surface



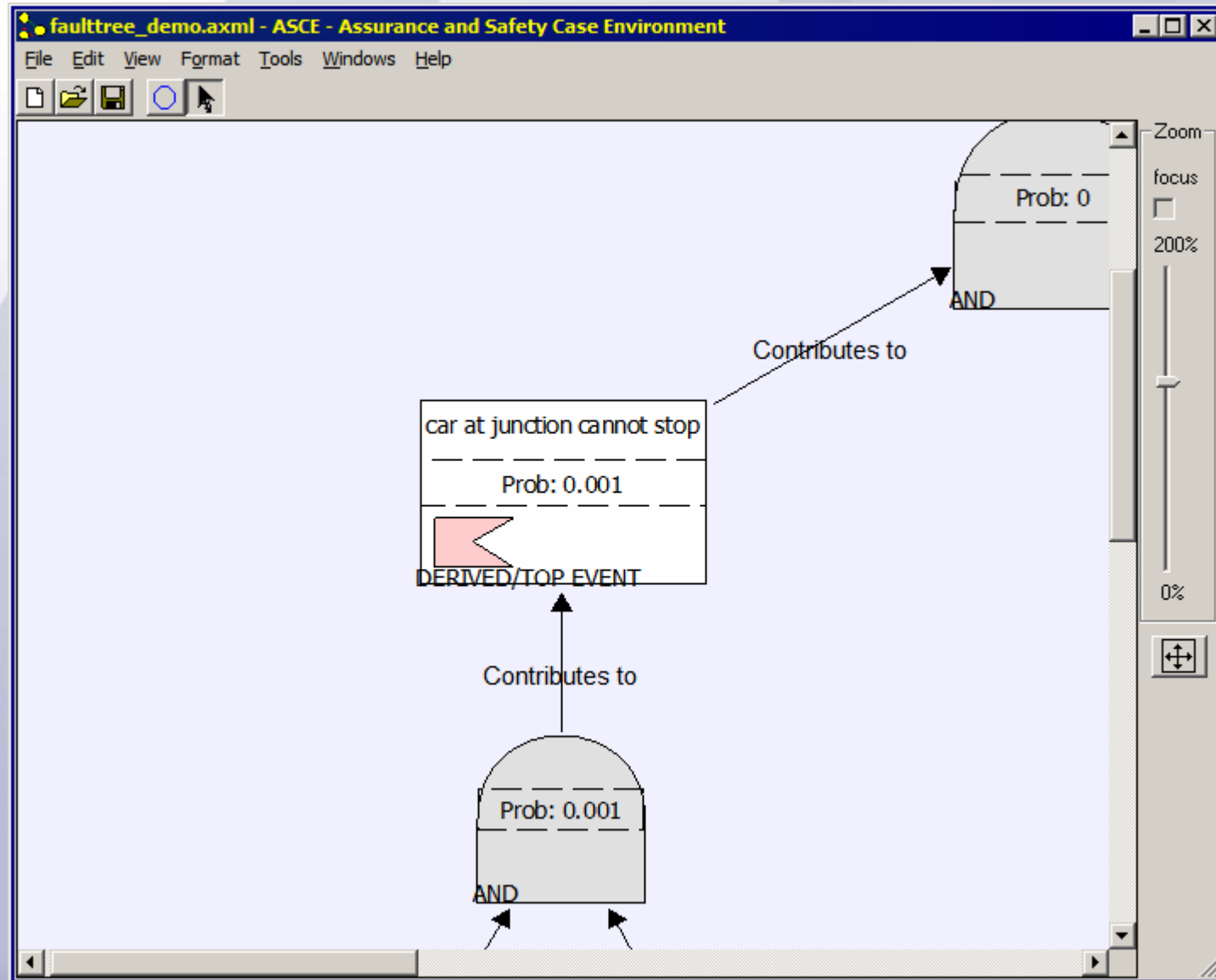
Display engine

- Application to simple fault trees
- ASCE already has simple schema and plugin for fault trees
 - But shapes are “non standard”
- New display engine allows us to have better shapes
 - E.g. AND/OR Gate
- Can also display node probability on the shape
 - Avoids previous workaround/hack of using node id
- Following screenshot also shows use of polygons as node decorators
 - Illustrates flexibility of new display engine

Fault tree example



Fault tree example



Other extensions

- ASCE files have embedded images in them
- Would be nice to be able to show these images on the canvas
- Could be nice to show logos of organisations/stakeholders in a safety case
- Possible performance impact
- Preliminary implementation

Still to be implemented

- To allow for wider range of ASCE based applications
- We aim to extend visual display engine further
 - ASCE links - curved lines – more beautiful?
 - Plugin based decorators as well as schema ones
 - “Hot spots” on node surface
 - Events caught by plugin handlers
- Working better with the plugin engine
 - Clicking on “hyperlink” on a node starts a plugin action
 - launches an external file
 - Custom editor managed by a plugin
 - Contextual feedback or analysis
 - ...

Still to be established

- Acceptable performance
 - ASCE v3 is efficient and scalable
 - Some networks have hundreds of nodes
 - Does new display engine add an acceptable performance overhead?
 - Maybe not all aspects will make final release
 - E.g. images on canvas

First power drop

- Will have partial implementation of new display engine
 - New decorators
 - One or two sample schemas to illustrate
 - Fault tree
 - One other
- Will have bugs in it
 - So don't use for any real work
- Contact asce-support@adelard.com if you are interested in trying the power drop when it is released



End of slides