



ASCE v4 Display Engine

Painting and Decorating in ASCE

Luke Emmet — luke.emmet@adelard.com

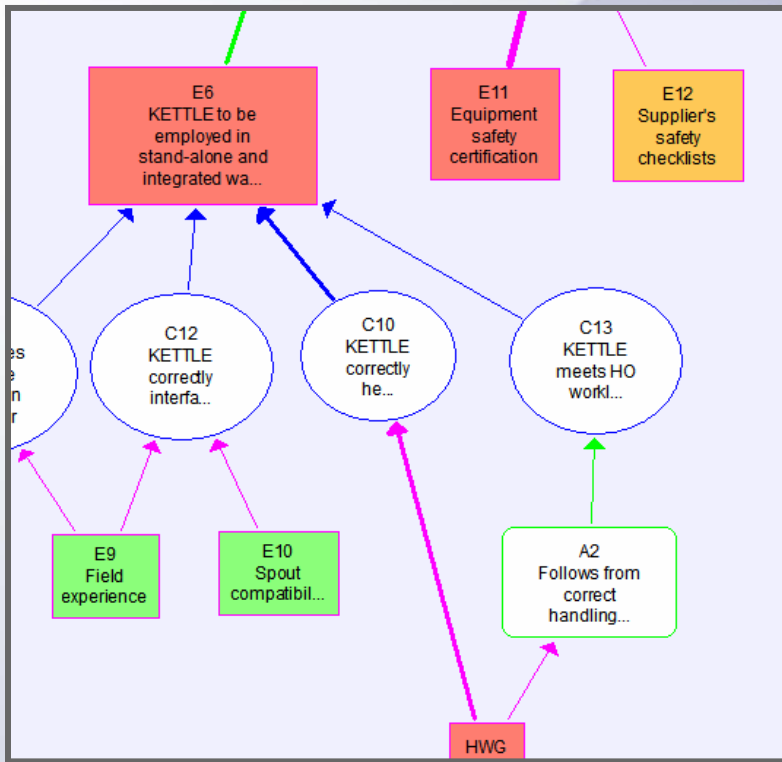
19-May-2009

Overview

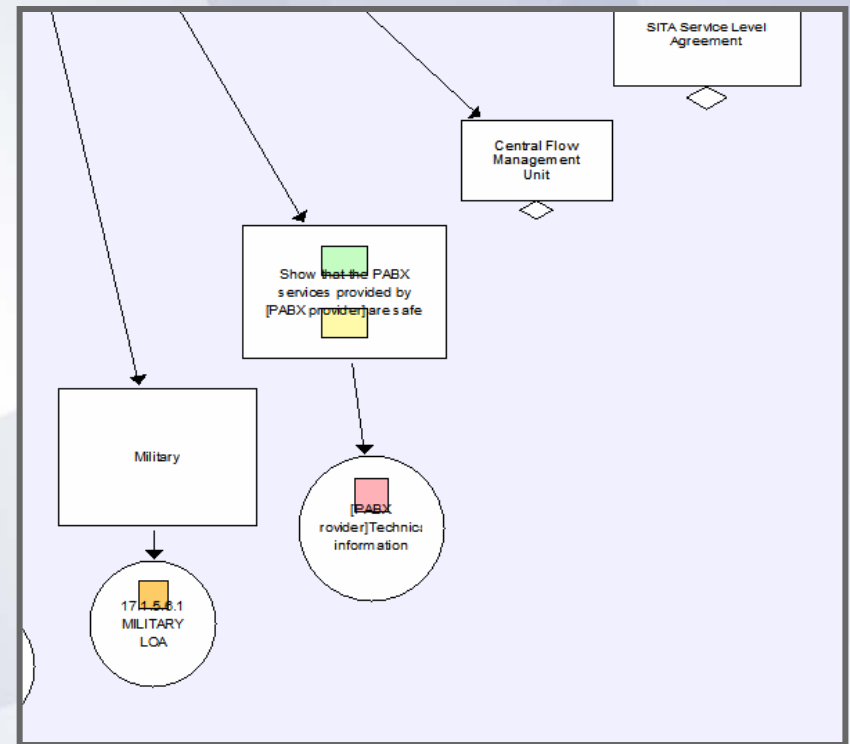
- Background and motivation
 - Building on ASCE 3.5 display engine
 - Motivating scenarios
- Display engine
 - Design and concept
 - How does it work
- What does it mean for me?
 - Benefits for users
 - Benefits for developers
- Progress to date
- Demo
- Questions

Background and motivation

- ASCE Display engine has served us well over the past few years
- Flexible enough to implement a wide range of notations
 - Claims Arguments Evidence
 - GSN
 - Bow tie analysis
 - ...
- Main advances in the area of node decorators
 - GSN decorators (to be developed, to be instantiated)
 - Confidence/Traffic lights
 - Spectrum 1 and 2
- But not powerful enough for some things we want to do now



Traffic lights



Other decorators

Requirements

- For v4 we want to extend display engine as a key requirement
 - To support wider range of user applications
 - ASCE as a “meta-tool”
- Including as a starter:
 - GSN “modular extensions”
 - Better fault trees
 - ECF (events and causal factor charts)
- Innovative applications we might create
 - As core ASCE services
 - As commercial addons
- Innovative applications you might create
 - Organisation-specific solutions
 - New applications altogether

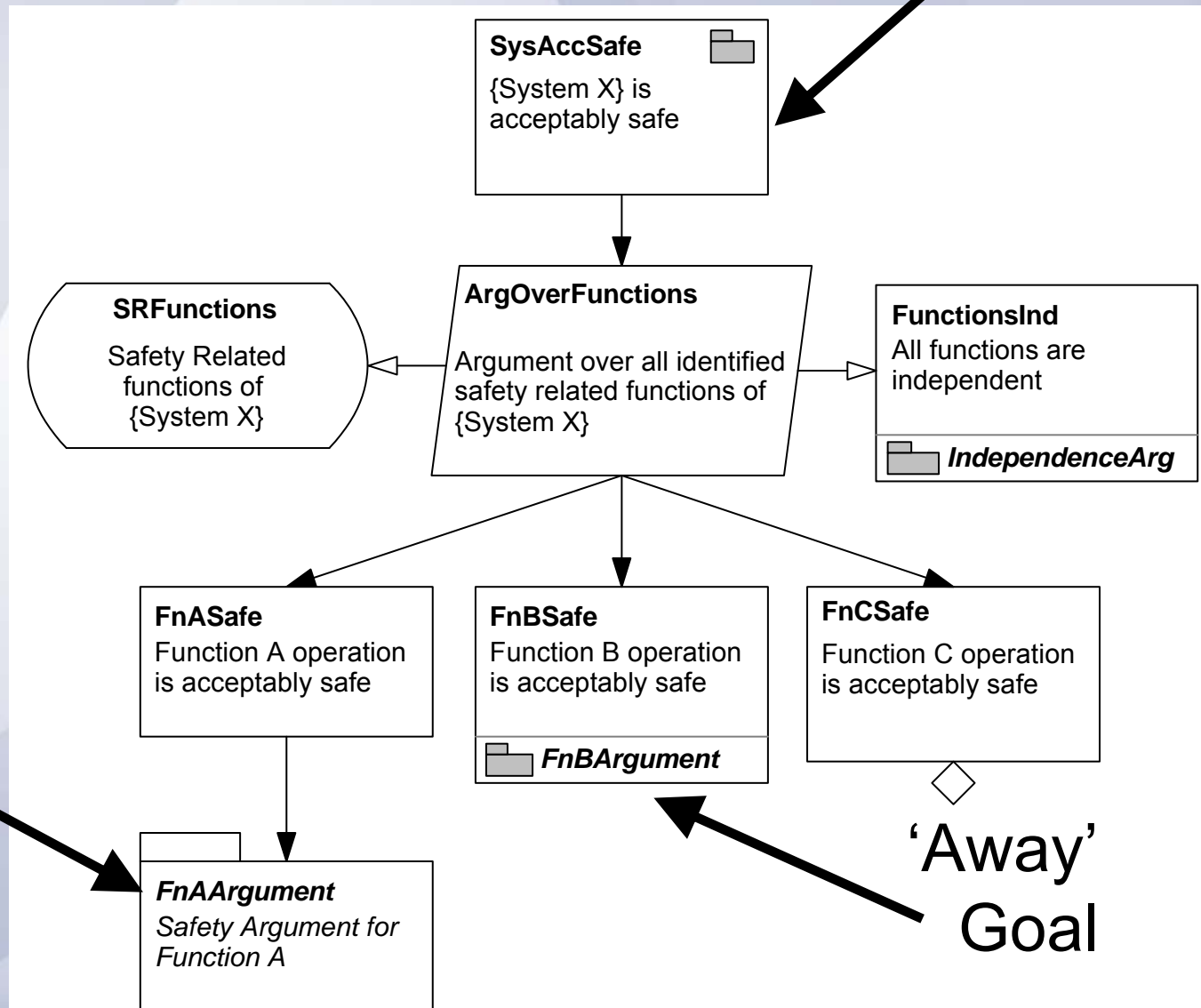
Scenario 1 – Modular GSN

- ASCE is the most widely used application for developing GSN structures
- Modular extensions have been published, but little existing tool support
 - York visio addin
 - Very little commercial grade support
- Some GSN users very keen
- Combination of enhanced display and behaviour requirements

Modular GSN extensions

Public Goal

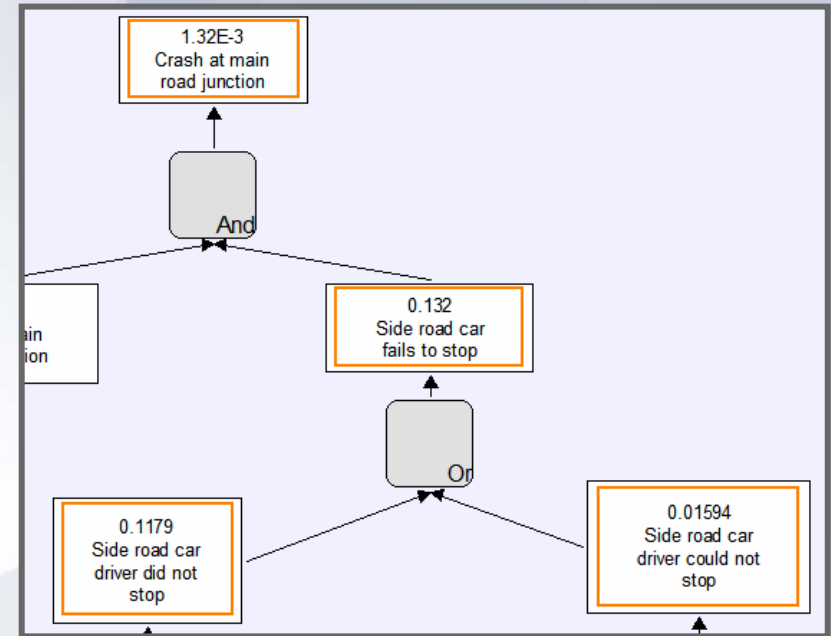
What would be the ASCE implementation of these extensions?



Module Reference

Fault trees

- ASCE 3.5 implementation is functional but visually non-standard
- Plugin has to use a hack to display probability of each node
 - Puts it in id to have it displayed
- Would be nice to use more standard shapes
- Customer interest is growing
- A test bed for the new ASCE v4 display engine





ASCE v4 – a new display engine

Design and concept

- New display engine in ASCE v4 is well underway
 - Some aspects were demonstrated at last user group
 - Got positive feedback on this early prototype
- Overall concept is for more flexibility in ASCE applications
- Provide a richer visual language for
 - ASCE schema designers
 - ASCE plugin developers
- Every node can be a composite of different visual aspects:
 - Basic shape
 - Information from the node itself
 - Extra decorators added by plugins
 - A picture from ASCE picture collection [*intended, but tbc*]

How does it work?

- Developing the new display engine has required a revamp of new object model
 - A key enabler ASCE v4
 - Behind the scenes unless you write plugins
- Each node is built from a number of visual components, called *Decorators*
 - Decorators have shape, colour, text, position
 - Can be polygons
 - In layers
 - Can overlap
- Decorators can have events
 - Double-click on decorator and a plugin responds, e.g.
 - Underlined blue decorator: Launch a linked file
 - Open a custom editor
 - Display additional information
 - ...

Decorator layers

- From the bottom of the canvas these are

Layer 0. (TBC) – optional background picture

Layer 1. Data driven decorators, from the schema display rules

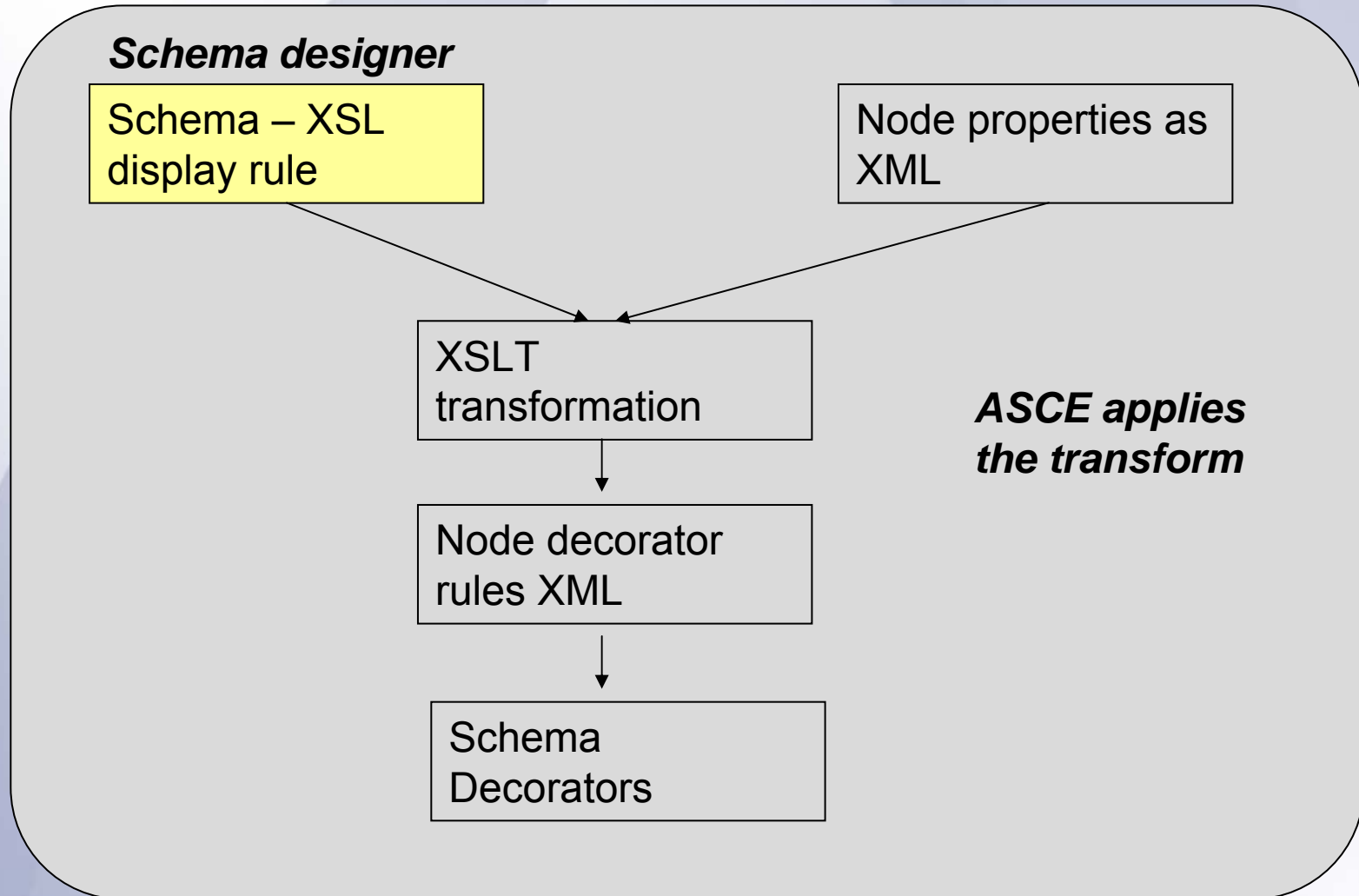
Layer 2. Extra decorators from plugins

- Layer 2.1
- Layer 2.2

Layer 1: Schema Decorators

- The schema has rules that determine the decorators, e.g.:
 - Node type -> outline shape, draw colour, position on node
 - E.g. Claim -> blue ellipse
 - Evidence -> pink rectangle
 - Node id/title -> transparent decorator of its text
 - Status fields -> decorators on the node
 - E.g. on Fault tree schema, show probability
- Rules are defined in advance in the schema itself
 - But can have dynamic behaviour
- Intended for “inherent” display aspects of the notation
- Data driven
 - Change the data, and the decorators change
- Specified in the schema as XSL expressions

XSL display engine



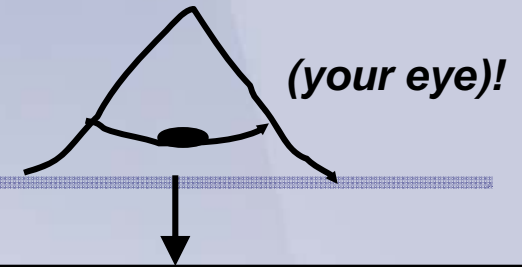
Plugin decorators

- Extra decorators from plugins
- Can be added to any node in layers
 - Each plugin can create one or more
 - Each layer can have many decorators
- Can be defined and created at run time
 - Plugins can listen for node change events
- Don't have to reflect data of the node itself

Highlighter_Plugin2.xml	decorators to denote user highlighted or not
Commenting_Plugin1.xml	decorators to denote external comments from PQR
Commenting_Plugin1.xml	decorators to denote external comments from XYZ
Commenting_Plugin1.xml	decorators to denote external comments from ABC

Putting it together

As you look down you see a composite picture



... (more plugin decorators)	
Highlighter_Plugin2.xml	decorators to denote user highlighted or not
Commenting_Plugin1.xml	external comments from PQR
Commenting_Plugin1.xml	decorators to denote comments from XYZ
Commenting_Plugin1.xml	decorators to denote comments from ABC
... (more schema decorators)	
Schema rule	other conditional aspects (e.g. "has external reference")
Schema rule	decorators to show fields from the node
Schema rule	decorators of the basic shape and node title
Schema rule	decorator for "traffic lights" (background colour)
... (more schema decorators)	
(possibly background picture)	

Benefits for users

- For day to day users of ASCE
- Wider range of potential applications
 - Modular GSN
 - Will follow as plugin/schema pair once ASCE v4 is released
 - OMG argumentation metamodel
 - Supporting ISO 15026 on Assurance Cases
 - Likely to find wide adoption
 - International standard
 - Process mapping
 - Content management
 - Fault trees
- More intuitive graphs
 - Show relevant information in familiar ways
- Plugins that help you do more stuff
 - Add comments
 - Highlight nodes
- You can build on your skills using ASCE
 - put them to a wider range of uses

Benefits for developers

- Your toolbox got bigger! 😊
- A wider range of applications can be developed in ASCE v4
- Schema development – wider range of options
- Plugins flexibility and power
 - Schema specific
 - to implement visual behaviour of the notation
 - E.g. events in modular GSN
 - ◆ Click on “away goal” – fluidly navigate to the remote file
 - Schema independent
 - E.g. commenting applications
 - ◆ where you don’t want to alter original file
 - ◆ Where the comments are outside ASCE

Progress to date

- Core object model mostly finished and implemented
- Display rules
 - Schema rules – complete
 - Plugin rules – initial implementation
 - Background images – still to be confirmed
 - Performance questions to be resolved
 - We don't want to kill ASCE's snappy graphical performance
- Events – to be done
- Curved links – partial
 - Will be a user option
 - Will work with manual link attachment hotspots
- Plugins
 - Display of externally collected comments – in the pipeline
 - Proof of concept of plugin decorators
- Demo applications
 - Fault trees
 - Modular GSN Proof of concept



Demo



Questions?

Thoughts? Challenges? Exclamations?