To appear in Proceedings of ACM Hypertext 2002 (HT'02), June 11-15, 2002, College Park, Maryland, USA

Graphical Notations, Narratives and Persuasion: a Pliant Systems Approach to Hypertext Tool Design

Luke Emmet & George Cleland
Adelard
Drysdale Building, City University
Northampton Square, London
+44 (0)20 7490 9450
loe@adelard.com, glc@adelard.com

ABSTRACT

The Adelard Safety Case Editor (ASCE) is a hypertext tool for constructing and reviewing structured arguments. ASCE is used in the safety industry, and can be used in many other contexts when graphical presentation can make argument structure, inference or other dependencies explicit. ASCE supports a rich hypertext narrative mode for documenting traditional argument fragments. In this paper we document the motivation for developing the tool and describe its operation and novel features. Since usability and technology adoption issues are critical for software and hypertext tool uptake, our approach has been to develop a system that is highly usable and sufficiently "pliant" to support and integrate with a wide range of working practices and styles. We discuss some industrial application experience to date, which has informed the design and is informing future requirements. We draw from this some of the perhaps not so obvious characteristics of hypertext tools which are important for successful uptake in practical environments

Categories and Subject Descriptors

H.5.4 [Information interfaces and presentation]: Hypertext/Hypermedia – navigation, user issues, The Adelard Safety Case Editor, ASCE; H.5.2 [Information interfaces and presentation]: User Interfaces – User-centered design; K.4.1 [Computers and society]: Public Policy Issues – Human safety

General Terms

Documentation, Design, Human Factors, Theory.

Keywords

Hypertext argumentation, graphical notation, safety related systems, safety cases, Pliant systems, usability, technology adoption, field experience.

1 INTRODUCTION

In this paper we present our experiences in developing a hypertext tool which combines graphical notations and narrative to support the user in constructing, evaluating and comprehending complex arguments.

The tool called ASCE[1] (The Adelard Safety Case Editor) [1], is being adopted within the Safety Industry, to support *safety cases*, which are essentially complex, structured arguments. Although rooted in argument development for the safety industry, ASCE has a wide range of applications, ranging from project management to informal hypertext development.

We provide the historical context for the development of the tool, followed by an explication of the tool together with its novel features from a technical and design perspective.

In our user-centred development process, we have sought to balance the requirements for supporting the development of rigorous safety arguments with the need to develop a genuinely flexible and pliant system as far as the user's experience is concerned. This raises interesting issues regarding user interface design and the complementary use of graphical notation and narrative in hypertext.

1.1 Notations for safety arguments

Modern safety regulation requires safety arguments (known as *safety cases*) to be developed and maintained as a primary means of communicating the safety requirements, safety management environment and supporting evidence for safety claims. In the UK, explicit safety cases are required for military systems, the off-shore oil industry, rail transport, civil aviation and the nuclear industry. Equivalent requirements can be found in other industry standards, such as IEC 61508 (which requires a "functional safety assessment") [2] and DO 178B [3] for avionics (which requires an "accomplishment summary").

A safety argument should [5]:

- make an explicit set of claims about the system
- provide a systematic structure for marshalling the evidence
- provide a set of safety arguments that link the claims to the evidence
- make clear the assumptions and judgements underlying the arguments
- provide for different viewpoints and levels of detail

Although safety cases are increasingly accepted and mandated for assuring critical systems, the traditional means of production – word processed documents with in-line graphics – has a number of shortcomings. Traditional applications have to be severely stretched for safety case development and the resulting documents are often cumbersome, and can be difficult to

[1] Evaluation copies are available for download at: http://www.adelard.com/software/asce. While ASCE is a commercial system, Adelard license it to academic institutes free of charge for non-profit teaching and research.

construct and review. Moreover, the structure of the safety argument itself is often lost in the volume of paper produced.

Toulmin developed a conceptual framework and graphical notation for representing the structure of an argument in the 1950s. Toulmin [6] makes a distinction between "claim or conclusion whose merits we are seeking to establish" and "the facts we appeal to as a foundation for the claim". Together with the notion of a "warrant" that the facts indeed support the claim, Toulmin developed the following basic notation:

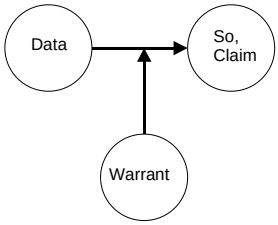

Figure 1: Toulmin's graphical argumentation motif

In this way, an argument can be constructed from the following elements[2]:

- a claim about a property of the system or some subsystem
- evidence which is used as the basis of the argument
- an argument linking the evidence to the claim, which explicates *how* the evidence supports the claim (e.g. statistical, logical argument etc.)

Since the evidence for a claim can itself be another sub-claim, or argument fragment, a generic graphical argument structure allows for hierarchical decomposition, with the general form of an *acyclic graph*, since a piece of evidence may support a number of claims. Cycles are problematic in that there is no grounded evidence for a cyclic structure, and nodes can be potentially self-defeating.

Following Toulmin's approach, more recent notations such as ASCAD [4], [5] and GSN (Goal Structuring Notation) [8], [9], with supporting methodologies have been developed for making arguments in industry. ASCAD uses a "claims-arguments-evidence" motif for representing argument structure (see Figure 2); GSN uses a similar "goals-strategies-solutions" form of construction. Existing hypertext systems that have adopted elements of Toulmin's schema include AAA [10] and more recently Aquanet [11].

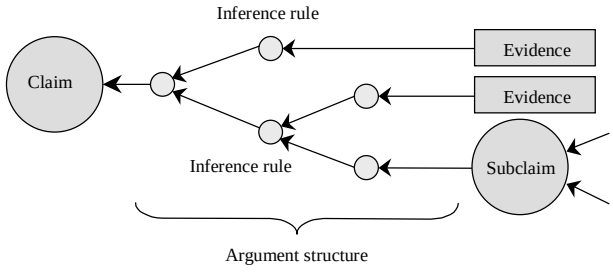

Figure 2: A generic graphical argument using a "claims-arguments-evidence" structuring motif

[2] This formulation is for arguments supporting claims *about* systems. Other domains for rigorous argumentation would assert claims about that domain. In the limit, Kolb [7] conceives of tools that could even enable argumentation about its own structures ("scholarly hypertext").

1.2 Integrating narrative and graphical notation into a hypertext argumentation approach

There are deficiencies in just using plain narrative, or a purely graphical notation. Pure narrative is critiqued earlier – such documents can be long, unstructured, and often do not bring out the implicit argument. Pure graphical notation can demonstrate links between argument sections and differentiate between different types of argument components, but without narrative there is no "meat" against which the soundness of the argument may be judged.

We propose that a hypertext argument is like software, in that it is "enacted" through reading the notation in conjunction with the narrative. At a macro level, a user "reads" and expands the graphical argument with narrative according to the structure, thereby recreating an overall narrative/set of utterances for the hypertext argument (e.g. "this node is the evidence for the 'design diversity' argument which supports the main safety claim"). The graphical notation allows the user to focus on particular structures and follow threads of supporting evidence. Similarly to software there can be bugs in the argument (e.g. a claim may be floating or unsupported, or a piece of evidence may be invalid). At a micro level, there is a need for standard narrative so that authors can explicate any necessary details about the argument, situate it in context, and make use of any existing narrative.

2 TECHNOLOGY ADOPTION ISSUES FOR HYPERTEXT SYSTEMS

Graphical hypertext systems exhibit a number of salient features which commend them as representational forms for complex information-based working:

- their design can make use of user's visual memory, embodying notions of place, closeness, shape colour and layout
- they provide a familiar form for representation (maps, flowcharts, graphs, schematic diagrams are common-place)
- simple, yet sufficiently expressive notations can be read as narrative which eases learnability and comprehension
- an appropriate domain-specific (or problem-specific) graphical notation can be "tuned" to express a meaningful set of semantic constructs (ranging from the informal to the formal [15])

Although a range of systems have been developed [16], [17], there have been some significant problems in their adoption as a technology outside of their research contexts. Conklin et al report that many systems "did not receive sustained use due to cognitive overheads and representational inflexibility" [14].

2.1 Usability

It is widely recognised that usability issues are critical in the design and adoption of software and hypertext technologies [18].

Usability advocates maintain that for systems to be adopted and meet the users' needs they must be developed with usability issues at the forefront of the design philosophy, rather than as an afterthought. Although software should support the user's goals, the interface itself should be "invisible" as far as possible. Norman [19], [20] refers to this kind of invisibility in the sense of having a "natural" interface and not occupying any psychological space. Weiser and Brown [21] suggest for systems to empower and support the users, application design

should allow fluid movement between the "centre" and "periphery" of the system, enabling local work in a wider context. For hypertext systems design, this implies the need to support the user in their comprehension of the wider hypertext structure in which local content is created together with fluid transversal mechanisms of those hypertext structures during authoring and browsing.

2.2 Pliant Systems

One form of general criticism of current software design and development practices is that the dominant ideology of software development is based on the belief that software should be written with all the uses formally characterised before development. But the system that the user receives is often overly rigid, not allowing for differences in working style or flexibility in use.

Henderson and Harris contrast this commonly held approach with what they call a "Pliant Systems" [22] approach, which aims to provide deep flexibility and "co-productive" style of support to the user. They suggest it is as if the current systems ideology develops systems according to an overly militaristic organisational model (the system is "locked down") in opposition to a market model ("chaos"). Pliant systems exist in the wide space between these two extreme perspectives.

Although there are not many examples of this approach in current systems[3], Henderson and Harris give an example from a "classical" organisational system – the bureaucracy – suggesting that its best invention was of the margin. A margin provides a place for user input that is *outside that which the system expects* – an "unformed" part of the system, providing a place for user comments, scribbles, notes, caveats and interpretations. "Post-it" notes play a similar role. Adding margins to the design of a system is thus a potential solution to creating more pliant use of rigid technology [22]:

> "Put a field called margin, or add to every field the ability to tag it, the ability to say 'Here is some more stuff'".

Another dimension to the Pliant Systems approach is to support a range of social practices regarding how people invent terminology and come to agreements on the use of terminology [22]:

> "People will invent terminology, they will come to agreements on certain kinds of conventions, other things will be left open and flexible"

This phenomenon is in stark contrast to the usual practice of software development whereby the "solution" is thought out in advance, then implemented in software. That people invent terminology and conventions relates to the experienced problems of representational inflexibility noted above. A Pliant System approach aims to honour these essentially *social practices.*

These experiences have also been borne out within the hypertext community; previous experiences in using semantic and navigational hypertext suggest that users engaged in certain classes of activity tend to avoid using overt structuring techniques in favour of more informal interpretations of collaborative and individual sense-making [23]. The development of spatial hypertext systems such as VIKI [25] is one response to these perceived problems of the imposed constraints that overt formality can impose on users and their working processes [24]. In such systems some linkage is left implicit and exists as an emergent function of spatial arrangement and properties (e.g. through spatial analysis).

2.3 Workplace integration

Workplace integration is essential for technology adoption. Although it might be seductive to think of a user spending most of their time using a novel application, the truth is that most of the time users are using other, established applications. Pragmatically, organisations may be unable to adopt any technology that does not fit into existing day-to-day work practices.

Hypertext provides a fundamental mechanism for enabling technology integration – the link – this is necessary but not sufficient for technology adoption. Other key aspects are:

- *Desktop and application integration* – users typically create much content using standard word processing (and other) desktop applications. Applications need to support import and export of text and graphics (from fragments to complete documents), in common formats.
- *Open standards* – applications should use existing open standards to lower barriers to long-term use and uptake. This reduces the risk of technology adoption for the user by protecting their intellectual investment.
- *Internet/Web* – in spite of the limitations of the web as hypertext, hypertext applications must export their structures for use on the web. Also users need to be able to link to standard Internet resources from within their content. This enables the results of hypertext applications to be widely disseminated, making use of "network effects" [26].
- *Paper* – paper is central to organisational and inter-organisational functioning. Being able to map hypertext documents on to equivalent printable version for a wide range of reasons (lack of on screen real estate, attaching onto an office wall, scribble on, deliver to customers and management) is a major technology adoption issue. Hypertext systems should therefore offer alternative linearisations of their structures for printing purposes.

3 THE ADELARD SAFETY CASE EDITOR

The Adelard Safety Case Editor (ASCE) is a hypertext tool for safety argumentation and hypertext development.

We adopted a user-centred design approach focussing initially on the tasks of safety case development. For us, this was possible, since we initially developed ASCE to support our own working practices[4]. This has ensured the benefit of a tight feedback loop between design and use; our internal customers were very much involved as the system evolved. As other users have adopted ASCE, we have greatly benefited from their use and feedback (see Section 4).

The design principles we have adopted are as follows:

- **Usability** – "quiet", fluid user interface using familiar and consistent interface metaphors (drag drop, standard behaviours, familiar menu layouts) fluid moving between

[3] Although it can be argued that aspects of the Internet meet some of the criteria

[4] Adelard's core business is in the assurance of dependable systems, including industrial safety case development and assessment.

centre/periphery, in the dimensions of the system (graphical, narrative, link traversal).

- **Supportive** – the system guides the user, and gives a positive lead in terms of argument creation (meaningful default link types, argument check). Rather than forcing the user to create "compliant" structures, the system supports divergence and subsequent convergence towards more "desirable" hypertext structures (according to the interpretation) through the application of hypertext structure checking (see Section 3.4).
- **Pliability** – pliant use of notation and technology for less "formal" system development (see Section 4.2) multiple schemas and user schema definition (see Section 3.6). User defined annotation fields and narratives act as a pervasive pliant "margin".

ASCE recreates some graphical structuring facilities that have already been implemented in other systems such as Aquanet [11]. Experience with Aquanet suggests that users avoided formal link structuring, preferring to use spatial arrangement of data to show association and to delimit different parts of the investigation.

Our experience runs somewhat counter to this. Explicitly typed nodes and links are crucial in correctly structuring valid arguments, and are valuable aids to users in correctly marshalling the components of the argument. This difference we believe is because the argument structures for safety cases have a stronger semantics that the type of information the Aquanet users were working with. Moreover due to a broad consensus on the need for safety cases, users are fairly well motivated to use argument structuring devices that are simple to use.

The use of typing does not unnecessarily constrain ASCE users however, as is automatically applied during construction, but may be subsequently broken by users. Network checking can subsequently advise on where structure type rules are violated.

ASCE augments standard graphical hypertext with a full-featured narrative editor (see Section 3.5) where users can construct informal and traditional narrative to augment the macro-level structure. This annotation space is not controlled by the system, although if the users add some structure to it, this can be used as the architecture for embedded link construction. This approach also means that the user is not overly constrained by the notation; they can step beyond it with narrative and standard document structuring to describe richer relationships and contexts not supported by the plain notation.

To support emergent structure, users can make use of standard untyped embedded hyperlinks in the body of any narrative created. These may be shown graphically if desired. Thus an implicit structure can emerge through references in which users have not committed themselves to some of the link types. Subsequently they may add explicit structure to reinforce the emergent structure, thereby creating an explicit hypertext semantics.

Furthermore, users are not compelled to take a formal interpretation of structures they create (e.g. if they turn off the visual display of node and link types and "abuse" the notation). This allows for the development of informal hypertexts with local or even personal interpretations (see Section 4.2).

3.1 Graphical editor

A graphical argument in ASCE is composed of a number of types of node, connected by directional links. The nodes are differentiated by shape and colour, the links by colour and width. Nodes in an ASCE network are created graphically, and links between them are typed according to the way in which the nodes support each other.

Users can turn off the display of link and node types for pliant use. Also, as they become familiar with the shapes and colours used, they do not need to see the node and link types explicitly in the display.

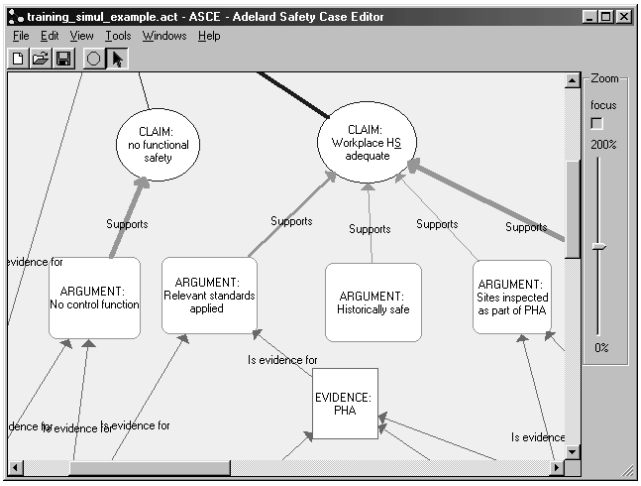

Figure 3: ASCE graphical editor showing node and link types (for the ASCAD notation)

ASCE supports multiple schemas for hypertext development. This means different schemas can be created according to the intended use of the tool, described further in Section 3.6. Each interpretation has its own status fields, check rules and display rules. These interpretations are abstracted from the tool itself.

ASCE currently supports two interpretation frameworks for hypertext arguments:

- ASCAD notation (Claims – Arguments – Evidence) [4], [5] – For example a *Claim* node might be "a sub-claim of" a parent *Claim* node, and an *Evidence* node would be "evidence for" an *Argument* node or another *Claim*.
- Goal Structuring Notation see [8], [9]. This has a larger set of nodes, the three key ones (Goal – Strategy – Solution) roughly correspond to the ASCAD nodes above (see Figure 8).

Alternatively, users may "abuse" the notation by turning off the display of node and link types to create hypertexts with a graphical map and structured documents beneath – useful for hypertext development generally (see Section 4.2). ASCE exports networks as HTML (optionally using SVG[5]) for the

[5] Scalable Vector Graphics – an XML standard for vector graphics. http://www.w3.org/TR/SVG

network graphics. ASCE supports cut/copy-paste of structures – both within and across networks. This enables reuse of content and argument "patterns" [9], [12].

3.2 Node properties

ASCE defines properties for nodes as user status fields. These are used formally or informally for process management/collaboration support/ recording status of network components. For example the GSN schema for ASCE has the following properties for each node:

- *Requires development* – boolean
- *Requires instantiation* – boolean
- *Completed*—boolean
- *Resourced*—text field
- *Risk*—ordinal, 0-5

These fields are editable as the network is developed, to track status of the argument, and to identify areas which require further work, or which have weak arguments.

3.3 Navigational features

As argument structures grow larger, there is a need to be able to see the view and navigate the network in different way e.g. to reduce clutter; to focus on a fragment of an arguments, to identify the strongest lines of argument etc.

Collapse and expand

ASCE implements a graph collapse/expand mechanism, which hides and shows fragments of the network according to the connectivity. The algorithm to hide a sub-graph for a node hides all and only those nodes that contribute to the currently selected node (see Figure 4 and Figure 5).

By collapsing a network at key points, the user can hide parts of the network that are "finished" or "acceptable", in order to see outstanding nodes and links. An inverse collapse (see Figure 6) is also implemented which allows a user to focus on a node and its relationships by only showing its subgraph (and hiding the rest of the network).

Zoom and focus zoom

The user can zoom in and out with ease using the zoom slider, which is always visible. This supports fluid movement between the centre and the periphery of the current focus area of the network. Focused zoom ensures that a selected node is always visible during zoom, so that the user can see its position in a wider context.

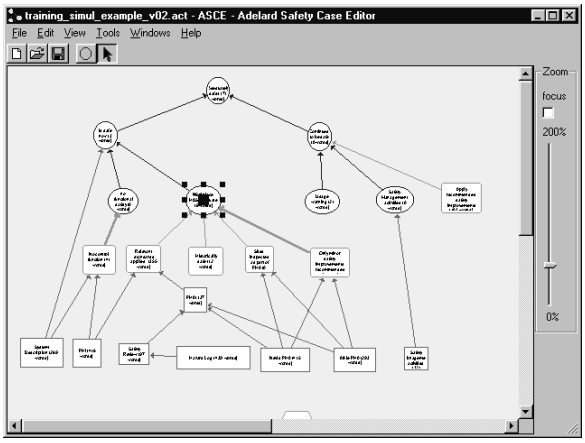

Figure 4: Graphical editor with node selected

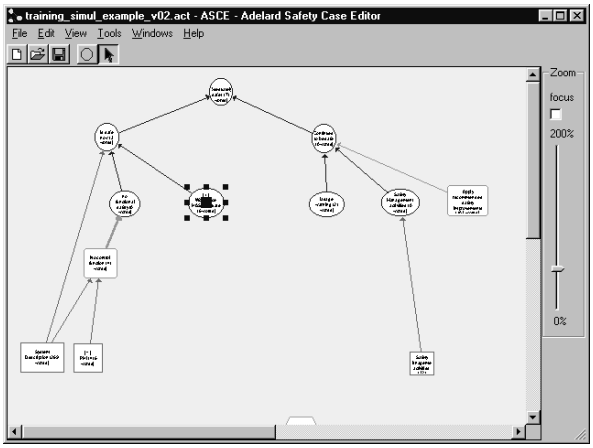

Figure 5: Network collapsed away beneath the selected node

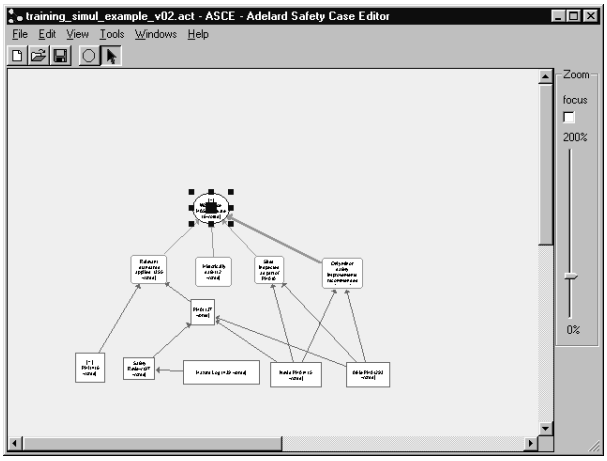

Figure 6: Inverse collapse – only showing sub-graph beneath some selected node

Fragment manipulations

Fragments of the graph can be selected for further actions such as dragging to a new location, forming the basis for a new "View" (see below), adding to an existing view, copy/pasting to the current (or another network).

Filtering

The user can interactively graphically filter the current network by node type and/or link strength. (see Figure 7). This can be used to show the main "thrust" of an argument. In other schemas it could hold a different interpretation (e.g. users could decide to use the link strength to denote criticality).

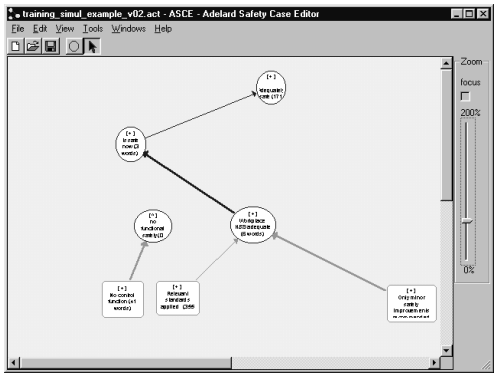

Figure 7: Graphical filtering the network based on node type and link strength

User Views

A view is a subset of the nodes with a persistent secondary layout. This is useful for showing and exporting key fragments of a large argument. It also can be used to simplify the presentation, navigation, export and printing of subset of nodes.

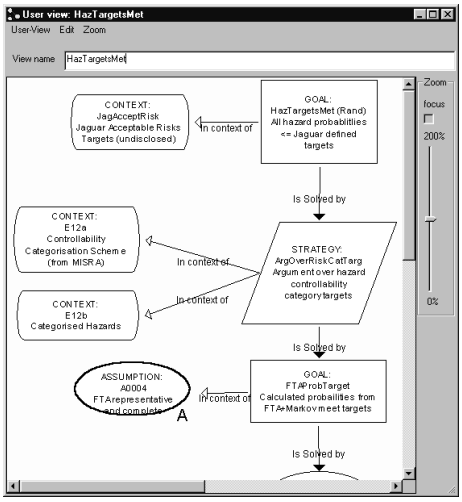

Figure 8: User view containing a GSN argument fragment

Nodes can be deleted from, or added to, a view, and the view geometry may be changed to improve presentation, although the connectivity is the same as the main view. Layout changes made in a view do not result in any changes to the main network.

3.4 Hypertext structure checking

Rather than forcing the user to create "compliant" structures, the ASCE supports divergence and subsequent convergence towards more "desirable" hypertext structures (according to the current interpretation) through the application of hypertext structure checking. This can be thought of as "hypertext syntax validation", somewhat analogous to syntax checking done by compilers/interpreters for computer languages. For example the network can be examined for any claim without supporting evidence. Also circular arguments can be identified. These check rules are defined on a per schema basis (see Section 3.6) and are expressed in XML. Example rules include:

- "floating" claims – claims with no support
- networks having more than one top-level goal
- network circularities – these can be problematic if the user wants to employ a formal interpretation of the network as an argument structure

There is also an embedded link check facility to check and identify that all of the embedded links are valid (i.e. both the destination node and the sub-heading in the destination node exist for each link).

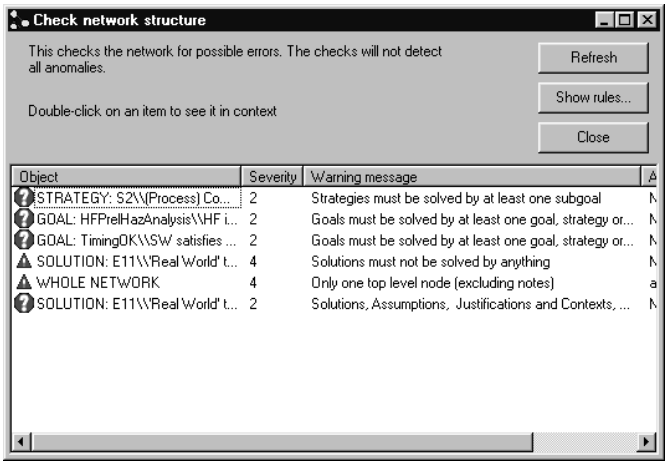

Figure 9: Network structure checking for a GSN network

3.5 Node Editor

Each node in an ASCE network has a narrative field which is a structured document in itself. The HTML based node editor is simple to use and allows users to import text from standard desktop applications. It supports text formatting, stylesheets, images, tables and heading styles to show the logical structure of the text in the node. This enables the argument to be correctly exported as a collection of HTML files.

Our concept for the Node Editor was to create an easy to use text-processing application supporting a useful set of document formatting functions without overloading the application with too many unnecessary features. Users can add embedded links to headings in any node in the network, any URL, or any local file. Authors can use this deep linking to create fine-grained hypertexts. Figure 10 and Figure 11 show the act of viewing (or editing) an embedded link. The user clicks on the embedded link, and the viewer/browser opens in the current editing context, enabling the user to select any other heading in any other node in the network, without switching out of the current editing mode.

The left-hand pane of the Node Editor contains a representation of the logical structure of the current node (see Figure 10). This view has five main functions:

1. to provide an overview of the structure of the node, as defined by the user. This augments the peripheral view of the current content during editing in a familiar, recognisable way.
2. to provide the architecture for fine-grained linking within the current network. Embedded links may point to any node, or heading within that node (see Figure 11)
3. to allow for rapid document navigation to specific sections
4. to allow sections of the document to be reordered
5. to cut/copy sections of the document

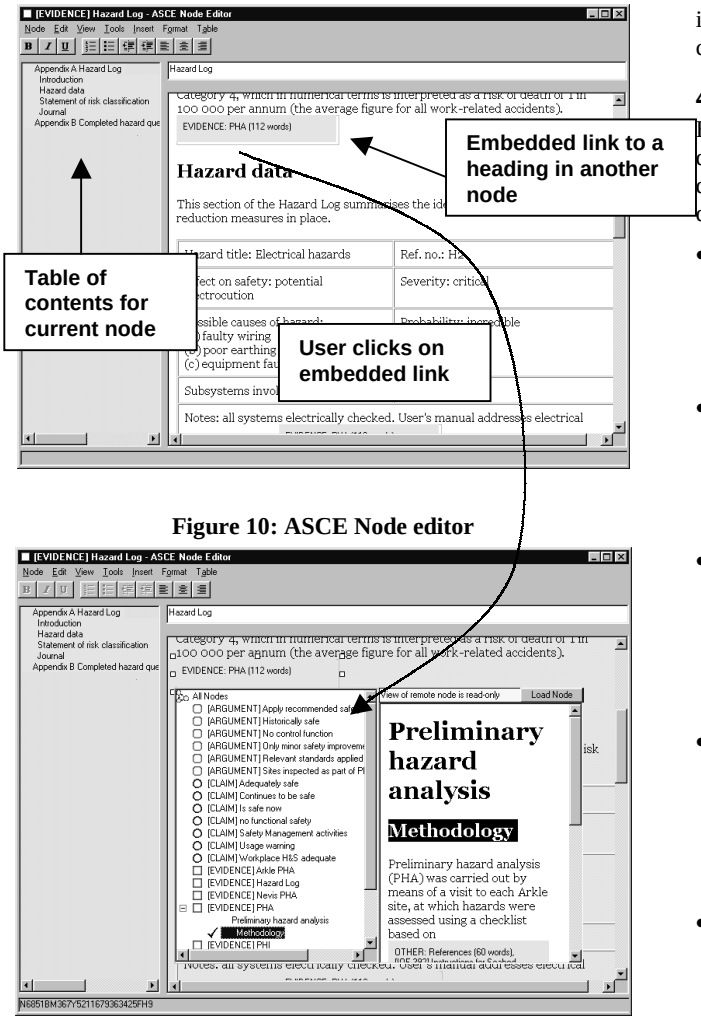

Figure 10: ASCE Node editor

Figure 11: ASCE Node editor showing inline link viewer/browser widget

3.6 Schema definition

ASCE supports the definition of domain specific schemas to implement different notations and support for a way of working with each notation. Schemas are defined in XML and have the following components:

- Schema Name
- Node definitions – node type, labels, shapes, colours, default link type, status fields – a collection of user-defined attributes for nodes
- Link definitions – link type, default direction, display text
- Display rules – visual modifications of the node according to values of the user status fields
- Check rules – desirable and anomalous structures to be checked for. Each rule has a severity, guiding those users who wish to use the check rules in deciding what (if any) changes should be made

4 EXPERIENCE IN USE

Experience and feedback from the ASCE user community has strongly driven the development of ASCE's functionality. In this and following sections we will provide examples from the range of applications on which it has been used, and the improvements to the tool which have come substantially from customer demand.

4.1 "Proper" use of ASCE

By this we mean use within the original vision for the tool – developing arguments, usually, but not always, in the dependability and safety domain – as opposed to use in different ways or domains which we document later.

- *Classical equipment or component safety case development*. This includes developing safety cases for: tracked vehicles; positioning and navigation systems for use in military or emergency vehicles; nuclear assay software; military air traffic control systems.
- *Aircraft safety evaluation and justification*. The UK military aircraft sector has a record of using of GSN to develop and document safety and operational arguments. ASCE has been used in several projects covering both fixed- and rotor- wing aircraft.
- *Legacy safety evaluation*. Increasingly safety analysis has to be conducted and documented for equipment, systems, or processes already in use in the field. ASCE has been used to consolidate existing safety material relating to such entities (often by hyperlink out to legacy documents), and to build new safety arguments.
- *Argument evaluation*. As discussed in the introduction, safety, and other, arguments are often delivered as text files which have little explicit structure to their argument. By "pouring" the text argument into ASCE, one can tease out the implicit argument, often in the process exposing weaknesses or areas of concern in the original document.
- *Large installation safety case development*. ASCE is being used to develop a safety case template for a major nuclear facility. This is a hierarchical document with many layers and component sub-arguments, each of which may themselves be complex arguments, or may be further decomposed.
- *Standards compliance*. We have experimented with several standards (e.g. IEC 61508, MoD 00-56, UK Railway Safety Case Assessment Guidelines), and developed prototype 'workspaces' which both encode the standard, and support documentation of compliance with requirements of the standard.

4.2 Pliant use of ASCE

Here we document some of the different ways that ASCE has been used in a pliant way (e.g. by developing alternative interpretations of the notation). This also illustrates the flexibility of hypertext systems generally.

- *Help systems*. The first pliant use of ASCE, was to write the help system for ASCE using ASCE. The node types are not used formally, but still provide a structuring facility for content and an overall map.
- *Note taking and brainstorming*. ASCE can be used to create "mind maps" or for general purpose note taking
- *Presentations*. The user-selected cascading style sheet for ASCE can reference font style and sizes appropriate for projection, rather than reading. So ASCE itself, or its HTML output can be used as a presentation tool.
- *Project management*. We currently use ASCE to support our ISO 9001 Management System. While not having the resource allocation and scheduling capabilities of

traditional project management tools, ASCE complements these tools well by providing a 'picture' of a whole project, allowing all project documentation from proposal through deliverables, reviews and progress reports to be accessed from a single co-ordinating document. Figure 12 shows a snapshot of a project currently in progress. This shows some of the main deliverables, meeting notes, review nodes etc, with an approximation of time running horizontally left to right. HTML export means that snapshots of a project can easily be shipped to project workers, clients etc.

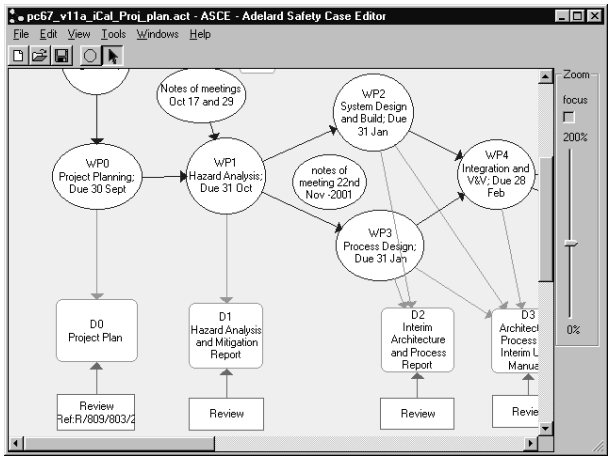

Figure 12: Fragment of a "Project Management" ASCE network.

4.3 The benefit of field experience

Field use has provided much feedback to us on both positive and negative aspects of ASCE. The original concepts behind ASCE as evidenced in the first versions have been vindicated – the tool is capable of expressing rich safety arguments. However, practical use of ASCE has helped us to understand a range of improvements which when implemented have supported effective deployment into existing operational environments. Some of the improvements we made were strictly functional, other were to improve the architecture of the system which has substantially increased our flexibility for future development.

Multiple schemas

The original ASCAD implementation was directly coded. In implementing the GSN version, rather than develop a further coding a schema interpreter was developed. The schemas are coded in XML, and the underlying code for different schemas is common. This allows us to develop new schemas as new application easily and quickly.

Structure checking

As indicated earlier, one of the key usability features of ASCE is the "pliant" environment where the user can "break" the rules of the notation being used. This avoids application behaviour which unreasonably constrains the user, and allows them to create networks flexibility, and with high productivity. This flexibility is controlled through checks on the network (see Section 3.4).

Reconciling hypertext systems with traditional approaches

While the facilities of a hypertext system may appear attractive, in practice they are not deployed as stand-alone applications. They usually have to integrate within an existing environment, which has existing infrastructure, procedures, and reporting and management structures. Early versions of ASCE, while elegant stand-alone tools, had some limitations in the quality of presentation and had primitive printing capabilities. This inhibited the conveyance of information out of the immediate field of use. Some considerable effort has gone in to improving these and similar "cosmetic" features, which, while not affecting the semantic content can impede the effective reception of output from the tool.

Status information

In development of a network from first concept to a mature document, the nodes will evolve through a number of different states. It is important that the user can record the current state of a network element, particularly in large networks or where there may be multiple authors. We have implemented a number of 'status fields' for each node which can be used to record process and project information.

Multiple views on to the data

Users very quickly started generating very large networks, some with several hundred nodes. While the filter/collapse facilities described above supported localisation and focussing during development, presenting the argument as sub-arguments proved problematic. To solve this we implemented user views and table views. These are described in detail above, but they supported the user in analysing the current state of development of their network, and in presenting the network in manageable fragments. Figure 8 shows a user view of a GSN fragment.

5 FUTURE DIRECTIONS

With the current release we have a mature product which is proven in practical use. The next stage is to develop facilities which will enable use on enterprise wide applications, including distributed working, and modular safety case development. In addition we are considering ways of providing stronger semantic interpretation across networks. This will involve considerable investigation, including analysis of user processes. As can be seen from the immediate problems below many of these issues are general research issues for hypertext systems in general.

Inheritance and impact analysis

A change in a piece of evidence may invalidate (or at least call into question) a larger fragment of an argument. We plan a simple inheritance function for certain type of status. For example, when a component is changed, its completion status is flagged as false. This then flags all immediate upstream nodes' completion status as false. This effect percolates up the tree until reaching either the top node, or a barrier level.

Browsing and importing data in other applications

We have already implemented the facility to open any arbitrary accessible document or hyperlink from within an ASCE network. There is however a need to import more fine-grained data into arguments. For example a safety case may appeal to information stored in a separate hazard log application. We plan to implement a facility to browse certain classes of application to identify data items or structures, import them to an ASCE network, and maintain a link to the data source. This will allow the data element to be updated easily. We will also investigate a data ageing check where all such accessed items are validated as current and providing an option to update if not.

Multiple rendered nodes

A piece of evidence may be used to justify more than one branch of an argument (see Figure 4). In simple networks this is accomplished by having more than one link out from a node. However, in large arguments this can result in excessive visual complexity. We plan a facility, perhaps similar to the approach in Aquanet [23], to support multiple rendering of the same node. There are difficult design decisions here, concerned with authority control for editing, and modular argument development.

Modularity/distributed argument development

While some use of ASCE has been for small arguments developed by one person. Considerable use has been on large-scale projects. For example a whole aircraft safety case may involve 50 person-years of effort spread over a period of 2-3 years. This clearly involves considerable concurrent working, and subsequent consolidation of material. In practice it also involves initial high-level argument development, followed by delegation (and possibly sub-delegation). It is therefore important to be able to decompose an argument structure, delegate authority for further development of a fragments, then re-link those fragments back into the main structure. We have developed some rudimentary tools to support this (copy/paste, external link to subsidiary ASCE networks), but these processes require very close management.

We plan development of more sophisticated facilities to more easily manage this process. However there are technical and process complexities that will require detailed analysis before an acceptable solution will be implemented.

User defined schemas and check rules

These have already been described above as a function we use for developing new schemas and network checks. However in their current instantiation they can only be used by experienced developers. We plan to develop simpler interface to support user-level definition of schemas and rules.

Formal model

We are developing a formal model of the underlying representation of schemas. To date the lack of the model has not been a major problem (apart from some feature interaction issues between collapse and filter). The only semantic analysis done so far has been via fairly localised check rules, and their interpretation with respect to the meaning of the schema is clear. Future check rules are likely to be more complicated and less localised, possibly spanning more than one network. Also, some of the features above such as modularity and inheritance will require interpreting data or status at one point in a network and inferring change in data or status elsewhere.

6 CONCLUSIONS

In this paper we have described our experiences in developing a hypertext tool for argumentation. In our approach we have sought to address usability and technology adoption issues from the start, in particular our design approach has aimed to design ASCE as a *pliant system*. This means that although designed primarily to support argumentation, the tool is sufficiently flexible to support a wide range of working styles and notations. The support for notation schema definition means that the utility is wider than the current application to hypertext argumentation.

The tool itself combines both notational/graphical support with a structured node editor for narrative creation. Both narrative and a structuring form are needed for creating and evaluating complex industrial arguments. We believe we have found a "sweet spot" in the balancing of more "formal" notation with structured and unstructured narrative. In part this has been due to the discovery of notations pitched at a particular level of abstraction; simple enough to be learnable and applied without deep technical expertise, yet structured enough to gain the benefit of creating an explicit structure. Moreover, the system itself can guide the user in creating and identifying "desirable" argument structures. This of course moves some of the responsibility for argument design into the design of the notations and argument schemas themselves.

In terms of our software development lifecycle, we have felt great benefit of being a user as well as developer of the system. This ensures we are addressing real needs and provides internal motivation for the project. In terms of requirements evolution, this enables us to achieve tight feedback loops in terms of usability and functionality as far as real users are concerned.

Having addressed technology adoption issues, we have been able to benefit greatly from field experience, and learned how users wish to use such technologies, both in a formal capacity (to develop better safety arguments) and also in an informal manner to solve a wider range of day-to-day problems.

As far as our users have been concerned, we have been occasionally surprised by their focus on "cosmetic" and presentation issues, but on reflection it is necessary to realise that a new tool is only on part of a complex chain of flow of information. What does a user do with their output of such a tool? Typically, it is passed on in a wider chain of influence and persuasion; professional looking output is therefore essential.

Looking ahead we see a wide range of potential applications. Certainly in the field of safety related systems, there will be a growing need to assure the safety of ever more complex systems in society. More generally, usable structured hypertext tools can ensure the adoption of hypertext technologies will occur.

7 ACKNOWLEDGEMENTS

Firstly we acknowledge the contribution of the ASCE team, in particular Robin Bloomfield, Tim Clement and Sofia Guerra. We also acknowledge the contribution from the ASCE user community, in particular Chris Caines. Lastly we would like to thank the ACM HT2002 reviewers for their helpful comments.

8 REFERENCES

[1] ASCE (Adelard Safety Case Editor) homepage <http://www.adelard.com/software/asce>

[2] IEC 61508-1, "Functional safety of electrical / electronic / programmable electronic safety-related systems CEI/IEC 61508:1998.

[3] RTCA/DO-178B Advisory Circular "Software Considerations in airborne systems and equipment certification"

[4] Bishop, P. & Bloomfield, R. A Methodology for Safety Case Development, Safety-Critical Systems Symposium, Birmingham, UK, Feb 1998

[5] Adelard (1998) ASCAD—The Adelard Safety Case Development Manual ISBN 0 9533771 0 5

[6] Toulmin, S.E. (1958) The Uses of Argument, Cambridge University Press, Cambridge, England.

[7] Kolb, D. Scholarly Hypertext: Self-Represented Complexity. In Proceedings of The Eighth ACM Conference on Hypertext, Southampton, 1997, pp. 29-37

[8] Kelly, T. Arguing Safety A Systematic Approach to Managing Safety Cases (1998). PhD Thesis, available at

<http://www.cs.york.ac.uk/ftpdir/reports/YCST-99-05.ps.gz>

[9] Kelly, T & McDermid, J, Safety Case Construction and Reuse using Patterns, Proc 16th Conf on Computer Safety, Reliability and Security (Safecomp '97) 1997

[10] Schuler, W. & Smith, J. Author's Argumentation Assistant (AAA): A Hypertext Based Authoring Tool for Argumentative Texts, ECHT'90

[11] Marshall C., Halasz, F, Rogers R, & Jansen W C. Aquanet: a hypertext tool to hold your knowledge in place. In proceedings of Hypertext 91 (San Antonio, Texas) ACM New York 1991.

[12] Bush, D & Finkelstein, A. Reuse of Safety Case Claims – An initial investigation. London Communications Symposium, University College London 10th -11th September 2001 <http://www.ee.ucl.ac.uk/lcs/prog01/LCS035.pdf>

[13] Emmet, L. Experiences of using open hypertext to support safety documentation from: The 5th Workshop on Open Hypermedia Systems (OHS) < http://aue.auc.dk/~kock/ OHS-HT99/Papers/emmet.html>

[14] Conklin, J., Selvin, A. Buckingham-Shum, S, Sierhuis, M. Facilitated Hypertext for Collective Sensemaking: 15 years on from gIBIS. 11th ACM Conference on Hypertext and Hypermedia (Hypertext 2001), pp123

[15] Tourlas, K (2001) Diagrammatic Representations in Domain-Specific Languages. University of Edinburgh, Dphil Thesis. Available at http://www.dcs.ed.ac.uk/home/kxt/thesis.ps.gz

[16] Conklin and Begeman, M.L. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Infromation Systems, 4, 6, 1988, pp. 303-331

[17] Halasz, F.G., Moran, T.P., and Trigg, R.H. (1987) NoteCards in a Nutshell. In Proceedings of ACM CHI + GI '87, Toronto, Ontario. ACM Press. pp. 45-52.

[18] Nielsen, J. (1993). Usability Engineering. Academic Press, Boston, ISBN 0-12-518405-0

[19] Norman, D. Making Technology Invisible: A Conversation with Don Norman. Bergman, Eric (Editor) (2000) Information Appliances and Beyond: Morgan Kaufmann (August, 2001 or earlier) Also available at <http://www.mkp.com/books_catalog/Ch1txt.htm>

[20] Norman, D. The Invisible Computer (2 October, 1998) The MIT Press; ISBN: 0262140659

[21] Mark Weiser and John Seely Brown. The Coming Age of Calm Technology, Revised version of Weiser & Brown. "Designing Calm Technology", PowerGrid Journal, v 1.01, http://powergrid.electriciti.com/1.01 (July 1996). October, 1996. <http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm>.

[22] Henderson, A. and Harris, J. Beyond Formalisms: The Art and Science of Designing Pliant Systems. Chapter 4 in: Klaus Kaasgaard Software Design & Usability: Talks with Bonnie Nardi, Jakob Nielsen, David Smith, Austin Henderson & Jed Harris, Terry Winograd and Stephanie Rosenbaum, Copenhagen Business School Press, October 2000. Also available at <http://www.pliant.org/Beyond-Formalisms.pdf>

[23] Marshall, C & Rogers, R. Two Years before the Mist: Experiences with Aquanet. Proceedings of ECHT'92, Milano

[24] F. Shipman and C. Marshall, "Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems", Computer Supported Cooperative Work (CSCW) , 8, 4 (Fall 1999), pp. 333-352.

[25] Marshall, C & Shipman, F M. VIKI: Spatial Hypertext Supporting Emergent Structure ACM Hypertext 1994

[26] Whitehead, E.J. Control Choices and Network Effects in Hypertext Systems, 10th ACM Conference on Hypertext and Hypermedia (Hypertext 1999)