Designing Interactive Instructional Authoring Tools


E. Cloete
School of Computing
University of South Africa
cloete@unisa.ac.za

P. Kotzé
School of Computing
University of South Africa
kotzep@unisa.ac.za

Abstract
Creating electronic learning environments that adhere to principles of usability of reusability, and which are also based on open standards such as XML requires expert skills. Available authoring tools are mostly aimed at instructional designers who have these expert skills. Higher education lecturing staff are increasingly expected to design e-learning materials whilst most of them are neither interested nor equipped to cope with this task. This paper discusses the design of an authoring tool to create interactive courseware with special emphasis on usability parameters. The significance of this tool is its enabling of non-technologist teachers to create electronic learning materials. The discussion considers the e-learning environment as a network consisting of several sub-networks. Sub-networks are composed of several network nodes connected via navigational constructs. Atomic elements of the proposed authoring tool are equated to network nodes making it possible to apply usability parameters to each node while at the same time considering the impact on the whole network.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User interfaces - user centered design, theory and methods
D.2.2 [Software Engineering]: Design Tools & Techniques - user interfaces

General Terms
Design, Human Factors, Standardization.

Keywords
Usability, reusability, e-learning, instructional authoring tools.

1 Introduction
Most lecturers are professional teachers and researchers with expert knowledge in their particular fields of interest - not necessarily including expert use, knowledge of, or interest in interactive software systems. Therefore their interaction with software, whether used to conduct research experiments, interpret data, document research outputs or create learning environments, should be as intuitive as possible. As part of their daily preparation, teachers create learning
environments where teaching and learning occur. Creating similar e-learning environments requires additional skills such as interaction with authoring support environment (ASE) software, and the preparation of additional electronic course notes. Although some teachers may be inspired enough to do this, few are actually able to create dynamic learning software. The available tools are mostly aimed at educational technologists who also have technical skills besides their educational design skills. Furthermore, maintenance of course software is highly demanding, and learning objects (LOs) that are not designed according to open standards thwart the re-use of these LOs to a large extent, and also prohibit the integration of LOs into central repository environments [2].

The focus of this paper is on the design & development of an authoring tool that can be used to create interactive courseware, with special emphasis on usability parameters. The focus is not on aspects of course design. The purpose of the paper is to describe a framework that adheres to important usability parameters, which can be used to design an authoring tool that produces interactive courseware with reusable LOs wrapped in XML tags.

In our discussion, we show that there are three basic requirements to consider when designing and developing an ASE tool, namely (1) reusable LOs, (2) the usability of the ASE and (3) the usability of the resulting courseware. Complexity is added to the design task of the ASE because none of these factors can be considered in isolation.

The layout of this paper is as follows: In Section 2 e-learning networks are introduced as a basis for understanding the discussion that follows. Section 3 is an exposition of the theoretical basis of the afore mentioned design requirements. These requirements are used in Section 4 to depict a design framework for ASEs adhering to re-use and usability principles. A discussion on an ASE prototype implementation follows in Section 5 and conclusions are drawn in Section 6.

2 E-learning networks
2.1 ASE and IC networks
The e-learning environment can be considered as a structure consisting of several network systems, where each such network system consists of a number of associated nodes connected via navigational constructs. Nodes are functionally defined units representing the content of the particular network. For example, a node can be an activity, a piece of information, an illustration, a video clip, et cetera. Complexity of the network can be reduced by categorising nodes into different node types. Typical node types include glossary-type, help-type, question-type, annotation-type, simulation-type, information-type, browsing-type and termination-type.
Navigational constructs depict the potential routes between different nodes within a network. It is desirable that the primary actors in a network should be responsible for controlling the flow of a particular session through the network. For example, each student should be able to determine his own learning path through the courseware. This implies that two users will experience the sequence of events in the same course differently.

As with node types, there are also different navigational construct types, allowing one to distinguish between types of relationships among nodes (Kotze, 97). Navigational construct types include contextual constructs, referential constructs, detour constructs, annotational constructs, return constructs and terminal constructs. Contextual constructs are invisible to the actor and are used as a route tracking mechanism of the the route that the actor has followed through the network. Referential constructs are observable navigational constructs embedded in the interface, allowing the actor to pursue different routes through the network - these are typically the links that appear on an interface. Detour constructs are similar in function to referential constructs, but instead of routing the actor through the main network, detour constructs connect the actor to another (usually auxiliary) network. Annotational constructs are also visible to the actor, allowing him to connect to an annotation-type node where he can make personal notes. As the name suggests, a return-construct returns from a diversion that was initiated by a detour or annotational construct, and finally terminal constructs indicate that the end of the current network has been reached.

Figure 1 Internetwork of e-learning networks
ASE is the acronym for Authoring Support Environment. IC is the acronym for Interactive Courseware.

Figure 1 depicts an internetwork of e-learning networks. Two main networks can be distinguished, namely the ASE\(^1\) network and the IC\(^2\) network. The ASE network is comprised of interactive software consisting of a number of goal-directed processes as well as a number of events driving these processes. The primary actors in ASE networks are the creators of courseware producing the IC network consisting of customised nodes and associated navigational constructs for deployment on CD, the Intranet or the Internet. The students are the primary actors in the IC network. In certain delivery modes, such as the Internet, teachers or facilitators are secondary actors in the IC network. The purpose of each of the main networks is to achieve a particular goal for its primary actors. In the case of the ASE network, the actor aims to create interactive courseware while the primary actors in the IC network intend to learn about particular field. As illustrated in Figure 1, there are a number of auxiliary networks attached to each of the main networks. The purpose of such an auxiliary network is to assist the actor in his interaction with the network. Auxiliary networks are not essential to achieve the primary goal of interaction with a particular main network. For example, in Figure 1, the ASE network has three auxiliary networks attached to it, namely the ASE glossary network, the ASE help network and the ASE tutorial network. Interaction with these networks may enhance focus and efficiency, but is not an essential element in achieving the actor’s primary goal. In fact, mature actors seldom interact with these three networks. In many cases there are overlaps between different auxiliary networks. These overlaps are often non-deterministic, which means that entrance from one network into another might imply that an actor could be trapped in a situation where the network never ends.

2.2 Integrated effect of interconnectivity between the main networks

The abstract framework of the e-learning internetwork that is explained in Figure 1 consists of a set of states as well as relationships between the states and allowable operations on these states. We associated two states with each of the network systems; namely an internal state and an external display state. The internal state of a network system portrays the position or condition of the nodes under development, as well as the variables that are required to control interaction with the author while creating and interacting with these nodes. For example, while a course designer creates a question-type node, values such as the type of node being created, special attributes for the particular node and conditions before the question or parts thereof (like its answer) may be displayed, are set. The external display state, on the other hand, is the perceived network interface with which the actor interacts. This state reflects parts of the (but not necessarily the entire) internal state. For example, the student who interacts with the

---

\(^1\)ASE is the acronym for Authoring Support Environment.

\(^2\)IC is the acronym for Interactive Courseware
ASE Network:
internal state

ASE Network:
external display state

IC Network:
internal state

IC Network:
external display state

(a)
(mapping)

(b)
(mapping)

(c)
(mapping)

(d)
(mapping)

Relationship

Figure 2 Mapping & relationships between e-learning networks

Figure 3 Input-output parameters of the main networks

question-type event observes the content and certain conditions that pertain to the question.

Figure 2, adapted from Kotze [4], illustrates the mappings between these internal and external display states within each network as well as the relationships among them. The interaction of the actor with the ASE network nodes is reflected in mapping (a) between the ASE's internal and external states. The internal state of the ASE network also maps to a resulting IC network state in mapping (b) that shows how the resulting IC network is reflected during the authoring process. Furthermore, the resulting state of the IC network maps to external perceivable rendering of the IC network as illustrated in (c). In the same way as for the ASE network, the internal state of the IC network maps to its external display state illustrated in (d). The dotted lines (i) and (ii) indicate that there is relationship between the different states in the ASE and IC networks.

As said before, our focus and interest lie in the ASE network. However, Figure 2 illustrates that it is impossible to consider the ASE network in isolation since the states in this network have a direct impact on the resulting IC network system.
3 Design requirements for ASEs

3.1 Reusable output components

Figure 3 illustrates the input-output parameters of the main networks, as well as their relation to one another. The primary actor interacts with the ASE network by defining the LOs using plain text, graphics, defining links, et cetera. The output of the ASE network becomes the IC network, accessible by the primary actor of the IC network.

Events in the ASE network can be organised into two main categories, namely the create category and the format category. In the former, events have the main purpose of creating content while in the latter, the content is formatted for presentation purposes. It is important that these two categories are kept separate so that reuse of the content is made possible. Mixing content creation and formatting might initially accelerate the generation of a course, but in the long run, it makes maintenance and reuse almost impossible. Probably the most common example is that of the hypertext media language (.html) files. These types of files are generally cumbersome to maintain and their reuse boil down to cut-and-paste actions. Furthermore, these files do not lend themselves to information retrieval - a basic requirement for reuse.

To enhance effective authoring, the actors in the ASE network must be able to reuse output components from one ASE network in another. Many software packages are available to create state-of-the-art learning environments (e.g. Framemaker, Quest, MS Frontpage). However, for a long time, requirements for e-learning and CBT focussed on the learner requirements (embedded in the resulting courseware) rather than the techniques and mechanisms to simplify the job of the creator and the maintainer of the courseware. Neither of these should be neglected if e-learning is to reach its potential.

3.2 Usability Parameters

Abstract principles for effective interaction derived from knowledge of the psychological, computational and sociological aspects of the problem domain can be used to direct the design and evaluation of a product from the outset. Three categories of general principles can be identified [3], namely learnability, flexibility and robustness. Some of these have been adapted from web-based applications [1]. The key to designing user interfaces for applications in a certain domain is to understand the important aspects of how work activities are performed in that domain and how one activity might influence the behaviour of other activities in the domain. The interface principles related to specific activities are important insofar as they help or hinder the users in achieving their goals. In the design of an ASE most of the general principles apply and several domain-specific ones contribute to the acceptance and effective use of the network. In
In brief, learnability refers to those features of the ASE directing novice actors in perceiving the intended interaction of the network, both initially and later on, in order to accomplish the desired outcomes. The learnability elements of interest in the design of the ASE include predictability, familiarity, generalizability, consistency and coherence. Flexibility refers to the number of ways in which the actor can interact with the network and therefore refers to the information exchange between actor and network. Usability parameters contributing to the flexibility of the ASE network include dialogue initiative, multi-threading, task migratability and customisability. The robustness of a network refers to the features that support the successful achievement and assessment of the goals. Usability parameters affecting the robustness of an ASE include observability, recoverability, responsiveness and task conformance.

One of our main interests lies in the relationship between the external ASE display and the resulting state of the IC network, i.e. how the resulting e-learning system is reflected at the ASE interface during the authoring process. We are also interested in the relationship between the external display of the ASE network and the external display of the IC network, i.e. how the student will perceive the resulting interactive courseware as reflected at the ASE interface during
the authoring process.

4 Design framework for ASEs
As shown in Figure 1, there is a strong relationship between the external states of the ASE and IC networks. The mapping from the internal state of the ASE to the external state of the IC network (via the resulting state of the IC network), also generates a relationship between these two entities. In fact, the design decisions made by the courseware creator during his interaction with the ASE are pertinent to the external display state of the IC network, and also to the way in which student actor interacts with the IC network.

Our purpose is to design a dynamic model for a usable ASE producing reusable course components. Owing to the specific type of relationship between the ASE network and the IC network, it is not possible to isolate the design criteria for ASEs from the design criteria of e-learning software. However, it is quite difficult to keep the design criteria for two different interfaces in mind when designing one particular interface. We provide a framework to integrate the design criteria of both the IC network with the “learner” actor in mind, and the ASE network, with the “teacher” actor in mind, into a single ASE interface. Figure 4 depicts the cyclic waterfall approach to this design problem as consisting of four phases.

Phase 1: Identify Learning units
The initial focus of the design is on the IC network, specifically on the content of a learning situation. This content might be a single lesson, a set of lessons having a common goal or subject theme, or even the entire syllabus. However, it is advisable that a single theme or lesson, rather than a group of lessons or the entire syllabus, be considered one at a time and carried through Phases 2 to 5 before commencing with the next theme or lesson as this will reduce complexities and enhance design focus. The best possible way of implementing this approach is then to iterate through the different phases for each theme or lesson. For the sake of simplicity, we call such a theme or lesson a learning unit (LU).

Focussing on the contents of a course, Phase 1 serves to identify all the LUs of a particular course. For example, let us consider an Object Oriented Systems Analysis course - typical LUs that might be identified include ‘Basic object-oriented concepts’, ‘Requirements gathering’, ‘Requirements Validation, ‘Dynamic modelling’, ‘Class modelling’ et cetera. As another example, say we consider a Computer Networks course, then typical LUs for this course include ‘Network Models’, ‘Network Media’, ‘LAN transmission equipment’, LAN Topologies, et cetera.
In essence, the ASE design cannot be either subject-specific or LU-specific at this level since the ASE should be generic in the sense that it assists in the design of any e-learning course irrespective of subject field. However, the above breakdown shows that we can formalise this phase as follows: For each course \( C \), there exists a set of LUs, \( \{ LU_k \}_{k=1}^n \) and the purpose of Phase 1 is to provide the ASE actor with a tool to identify such a set of LUs.

Once we have identified, isolated and formalised the different topics (segments) in the IC network, it becomes possible to apply specific usability parameters to the design of each of these segments.

**Phase 2: Topic identification**

The focus remains on the content of a learning situation. As suggested earlier, Phases 2 to 4 serve as a cyclic whole for each LU that has been identified in the first phase.

The emphasis of the second phase is on the identification of all the main topics of a specific LU. Returning to our previous example of the *Object-oriented systems analysis* course, let’s use the LU, ‘*Basic object-oriented concepts*’ to the identify main topics in this phase. Typical topics that might be identified for this LU include *objects & classes, class attributes, object & class relationships, methods, encapsulation, polymorphism, inheritance, generalisation & specialisation*, etc. As for our other example, the *Computer networks* course, let us suggest ‘*LAN transmission equipment*’ as a LU. Relevant topics that might be identified for this LU include: *NICs, repeaters, hubs, bridges, routers, switches and gateways*.

In the same way as in Phase 1, it is senseless to be topic-specific when creating a generic ASE. As before, we have to identify, isolate and formalise the different elements, forming small network segments in the IC network in order to apply the required usability parameters to each of these elements for the design of the ASE network. Using the formal approach from Phase 1: for each course \( C \), there exists a set of LUs, \( \{ LU_k \}_{k=1}^n \). Each learning unit \( LU_k \), is comprised of a set of topics \( \{ T_j \}_{j=1}^m \) that are embodied by the particular LU. The outcome of Phase 1, is a set of learning units, but only one of these learning units, say \( LU_k \), serves as input to Phase 2. The purpose of Phase 2 is then to identify the set topics \( \{ T_j \}_{j=1}^m \) associated with each \( LU_k \).

**Phase 3: Event declaration**

Teaching and learning take place through the interaction of both teacher and student in various learning events such as questions and answers, self-tests, information-transfer, exercises,
simulations, et cetera. In this phase, the focus shifts from the subject content to the methods of conveying the content. The emphasis in Phase 3 is to identify all the possible events that can be associated with each topic. Formally, \( \forall T_j \exists \{ A_{jh}\}_{h=1}^p \) that can be used to teach \( T_j \) effectively.

The importance is to capture the set \( A = \bigcup_{h=1}^p A_{jh} \forall j \) that underpins all possible learning events.

The purpose of defining such a union of event sets is to enable the course designer to select any event or events that he considers appropriate to render possible the learning of a specific topic. Each of these events becomes a node in the network segment of the specific topic. As mentioned in Phases 1 and 2, the different LUs constitute network segments, while the topics form smaller network segments within the LU network segments. The most common events encountered in a teaching and learning environment include information-transfer (describing a concept), question, example, exercise, self-test, simulation and a URI\(^3\) (link to an external resource such as a diagram, video clip, URL\(^4\), et cetera). This list can be extended, but it is advisable to guard against a too fine-grained categorisation of events, since this might introduce complexities with regard to the ASE interaction - especially where terminology unfamiliar to the common learning environment is introduced, focussing on computing technologies and terminology rather than on learning terminologies.

Each of the identified events has attributes that define the event and elaborate on it, making it flexible for use in different circumstances. It is at this point that the design focus of the IC network merges with the design focus of the ASE network. In the IC network, each attribute plays an important role in constituting a complete learning environment. However, in the ASE network, the presentation of each attribute and how the ASE actor can interact with it, contributes to the usability of the ASE network. For this purpose, we briefly formalise each event with its attributes. The basic format of all events is:

\[
E = \{\text{eventType, name, attributeList,}\ldots\}\bigcup N \text{ where }
\]

- \( \text{eventType} \) contains the specific type of event for example: ‘INFO’ and ‘QUEST’ identify information-transfer-type events and question-type events respectively;
- \( \text{name} \) is a unique name assigned to this event. Together, the \( \text{eventType} \) and \( \text{name} \) forms a unique combination identifying the address of a specific node;

\(^3\)URI: Uniform Resource Identifier

\(^4\)URL: Uniform Resource Locator (refers to a web address)
• The attribute list refers to a number of other parameters that are included in other events, which will subsequently be discussed; and
• \( N \) is a set containing zero or more navigational constructs.

Before considering the individual events, it is necessary to consider the concept of a compound event. In some cases it does not make sense to consider each event as a network node. For example, say it takes three pictures, four paragraphs, one simulation and a bulleted list to explain a particular concept. In this case, it is senseless to create nine network nodes to form the single concept (unless this is done so for re-use purposes). A compound event is defined as a set of events working together to construct a single network node, and is formally depicted as:

\[
E_{\text{compound}} = \bigcup E_i
\]

**Information-transfer Event**

\[
E_{\text{INFO}} = \{ \text{'INFO'}, name, descrip, items \} \bigcup I_i \text{ with }
I_i = \{ \text{rangeNo, itemNo, itemType} \}
\]

where
• \( descrip \) refers to the content of the information to be transferred;
• \( items \) indicate whether this information event includes a bulleted, numbered, or no list of items, taking possible values BLT, NUM, NILL;
• \( rangeNo \) refers to the number of the list, for example, if the information to be transferred consists of two lists, one bulleted and one numbered list, the first list would have a \( rangeNo \) of 1 and the second a \( rangeNo \) of 2;
• \( itemNo \) refers to the specific position of the specific item within its list;
• \( itemDescrip \) contains the content of this specific item.

*(Items are linked to a specific information-transfer event through the unique name of the particular information event.)*

**Question-type Event**

A question type event is defined to include maximum flexibility and therefore can be constructed by setting several attributes of which only a few are compulsory. Many of these attributes take a default NILL value, especially where a novice actor interacts with the ASE network.

\[
E_{\text{QUEST}} = E_{\text{compound}} \bigcup Q \bigcup O
\]

where
\[ Q_j = \{\text{range}, \text{no}, \text{show}, \text{cmnt}, \text{qst}, \text{ans}, \text{hint}, \text{crd}\} \]

\[ O_i = \{\text{opt}, \text{optValue}\} \]

where

- **range** refers to the question position whether it is a parent question or a child question. For example, if it is a child question of Question number 2, its range number would be 2.1, but if it is a subquestion of Question 2.1, its range number is 2.1.1 and so forth;
- **no** indicates the specific number allocated to the question;
- **show** is a value that the course creator can use to indicate whether or not the answer of the particular question should be shown, or when it should be shown.
- **cmnt** refers to any comments or notes that the course creator wishes to make about the question. These comments are not perceivable by the student actor, but only by the course facilitator;
- **qst** refers to the question itself;
- **ans** refers to the answer to the question, in the case of a multiple-choice question, \( \text{ans} = \text{nill} \);
- **hint** refers to hints pertaining to the question. These will be perceivable by the student actor;
- **crd** is the number of marks allocated to the question. A nill value indicates that there is no value attached to the question;
- **option** refers to the possible answer offered in a multiple-choice question;
- **optValue** is either True or False, depending on the relation of the **option** to its parent multiple-choice question. (The network will take responsibility for placing options in random positions.)

(options are linked to a specific question event through the unique name of the particular question event.)

**Exercise-type event and Self test-type event**
The elements of an **example-type** event are similar to that of an **information-transfer** type of event. A **self test-type** event is constructed from an **information-type** event plus one or more **question-type** events. In the same way, an **exercise-type** event is also constructed from an **information-type** event plus one or more **question-type** events. In each case, the **show** attribute determines at what stage the answer to the particular **self-test question** or the **exercise question**, is displayed. In other words, the **show** attribute is the factor determining whether or not the particular event is perceived (by the interacting actor) as a **self test-type** or **exercise-type** of event.
To remove redundancy, simple quick questions, quizzes, self-tests or exercises all refer to the same basic format. However, to enhance usability, it is not necessary that this be revealed to either the student or the course creator.

According to this discussion,

\[
E_{\text{exerc}} = \{ E_{\text{info}} \}_{i=1}^{n} \bigcup \{ E_{\text{quest}} \}_{j=1}^{m}, \forall n, m \in N, \text{ and} \\
E_{\text{selftest}} = \{ E_{\text{info}} \}_{i=1}^{n} \bigcup \{ E_{\text{quest}} \}_{j=1}^{m}, \forall n, m \in N
\]

**Annotation-type Event**

An annotation-type event allows the IC network actor to make personal notes and refer to other resources while working through the courseware. This type of event is defined to counteract the perceived system pre-emptiveness and also to increase the flexibility of the actor’s interaction with the ASE network. We formally define an annotation-type event as:

\[
E_{\text{ann}} = \{ '\text{ANNOT}', \text{name}, \text{memo} \} \bigcup E_{\text{URI}}
\]

where memo contains the notes of the actor and E_{\text{URI}} refers to a URI-type event.

**URI-type Event**

An URI-type event refers to an external source to be linked to the course. A URI-type event will seldom construct a network node, but rather form part of other compound events to form network nodes. However, since most nodes might require including reference to external resources, a URI-type event is an atomic unit that can exist alone, but will mostly be used in combination with other events. Also, an event, for example an information-transfer type event, might include several URI-type events. By keeping the URI-type events atomic, events in general are simplified. Formally, this event is defined as: \( E_{\text{URI}} = \{ '\text{URI}', \text{name}, \text{reference} \} \) with reference indicating the URI address.

**Simulation Event**

A simulation-type event can be used to enhance information transfer-type and exercise-type events. Since the inclusion of a simulation-type event often only implies a link to another URI such as an applet program, a simulation-type event is by default a URI-type event, but can also be constructed from other event types as a compound event.

**Auxiliary Network nodes**

As mentioned previously, the objective of auxiliary networks is to augment and articulate their main network. Therefore, the structure of an auxiliary network is functionally integrated with its
main network. Although an auxiliary network is an autonomous network, it cannot be designed to be entirely independent of, or in isolation from, its main network. For this reason we also consider the composition of three archetypal nodes from the different auxiliary networks: troubleshooting-type event, help-type event and tutorial-type event.

**Troubleshooting-type event**
A troubleshooting-type event provides the interaction opportunity to determine a possible reason for inexplicable behaviour of the underlying network, as well as measures to rectify or improve that behaviour. Formally, we define a troubleshooting-type event as:

\[
E_{\text{trouble}} = \{T\}_{i=1}^n \bigcup N, \text{ with } T = \{'TRBL', \text{name}, \text{descr}\}
\]

where the attribute *descr* describes the particular content of this troubleshooting-type event that is perceivable to the actor.

At this stage, it is necessary to define a secondary event type that is associated with troubleshooting-type events, namely remedy-type events. This type of event suggests possible remedies for a problem that requires troubleshooting. A remedy-type event never exists as an autonomous entity but is invariably linked to a troubleshooting-type event. Because there is usually more than one remedy-type event associated with a particular troubleshooting-type event, these events are connected indirectly through navigational constructs. Formally, we define a remedy event as \( R = \{'REMEDY', \text{name}, \text{fix}\} \), where the *fix*-attribute describes the particular content of the suggested remedy.

**Help-type Event**
A help-type event is considered to be an atomic event focussing only on one other network node or event type. Formally we define a help-type event as:

\[
E_{\text{help}} = \{H\}_{i=1}^n \bigcup N, \text{ with } H = \{'HELP', \text{name}, \text{descr}\}
\]

where *descr* refers to the particular content of the event that is perceivable to the actor.

**Tutorial-type Event**
A tutorial-type event consists of a number of navigational constructs referring mainly to information-transfer events, but can also include other types of events such as simulation-type or help-type events. Therefore a tutorial-type event is a compound event. Formally, we define
Phase 4: Navigational construct plotting

As explained before, navigational constructs describe the routing paths within a network. After the network nodes for a particular LU have been defined, network routes are designed and specified in a navigational table. In the IC network, this navigational table embodies a mesh of all logical (sensible) routes that can be followed through the network. Take note that the emphasis in route planning is on the IC network.

It is not feasible to maintain navigational tables that plot the complete path to every possible destination address in a network or to every possible address in an auxiliary network. Conceptually, it is important to use the principle of information hiding and allow routing decisions to be taken with minimal information available. Working from this principle, entries in a navigational table are stored in pairs of the format \((s,n)\), where \(s\) refers to the source node's ID and \(n\) refers to the next node's ID. A next-hop routing algorithm uses this one-step-along-the-path approach to identify the next node en route to the destination. As a first step in determining a route from source to destination, possible routes from the source are extracted. If none of these provide a direct link to the destination, entries that include the destination address are extracted next, and their sources are followed backwards until the shortest route from the source is determined.

To improve the retrieval capabilities from the navigational table as well as reduce the time it takes to make routing decisions, the navigational table is organised in a hierarchical scheme with direct connection entries first, same-network entries next, network-specific entries next, followed by a default entry last. The addition of a default entry reduces the possibility of routing failures, and can be a navigational construct that takes the user back to the beginning of the main network, or to the end of a specific section, et cetera. The logic on the navigational table directs the user to the default entry if no other suitable entry is found. This way a safe haven is created if the user traverses the network in an illogical way.

The design challenge for the ASE network is to provide the actor with suitable tools for route visualisation, route planning and route creation. The most flexible design architecture for this task is achieved by creating an environment in which the previously created nodes are perceivable to the actor - preferably depicted in a graphical form such as a metaphor or icon. Clicking on nodes to be connected creates navigational constructs. For example, say node A contains an information
transfer-type event and node C contains a question, and node A should be followed by node C. To plot the path and direction between nodes A and C, the actor selects a suitable button (e.g. 'Connect Two Units'), then clicks on the first link (A) followed by clicking on the second node (C). The underlying interface naturally assumes that the order is from A to C, but can easily be reversed (for example through a right click with a suitable option).

Some ASE actors prefer a hierarchical (sequential) design of the learning environment, while others prefer a lateral design. This is illustrated by the fact that some actors would define all the main nodes before defining accessory nodes augmenting the main nodes, while others engage in learning events of units that naturally belong together before moving on to new concepts. For example, say there are several concepts that have to be introduced in a particular LU, with many of these concepts to be explained through information transfer-type events only. However, some concepts require several types of events such as information transfer-type, simulation-type, self-test-type and help-type events. The first group of actors would typically define all (primary) nodes that present each of the concepts, before detailing any of the accessory nodes that help to create the complete learning environment. The second group of actors would typically decide on a particular concept to explain and then design all the augmenting nodes before moving on to the next concept.

For the second group of actors, their design task can be simplified by adding coherence to the interface. Coherence is increased when the ASE network records the design sequence of the actor. Depending on the order of the design, the external display state of the ASE network exposes the nodes within the natural groups for the route plotting exercise, with the option to disarrange the suggested grouping.

Observability of the ASE network is greatly enhanced by making the completed mesh graph perceivable to the actor. However, for large routing tables, the complete mesh may actually increase complexity of the external display state instead of simplifying learnability parameters such as familiarity and predictability. In such a case, flexibility parameters such as adaptability and adaptivity should be given special attention during the ASE network design, by making provision for the external display state to expose only a route cluster at a time instead of the complete mesh. Flexibility such as this can be established in the routing table by storing the route entries as triplets instead of pairs, with the third parameter indicating the natural cluster to which a specific route entry belongs.
Phase 5: Implementation of usability parameters

5.1 General implementation of usability parameters

Some of the parameters that enhance (or might cripple) the usability of the ASE network apply to the network as a whole, while others are applicable to individual nodes that have been identified earlier. We first comment on the parameters that apply to the entire ASE network.

It is possible to design a usable ASE network without considering all usability parameters to the same degree. For example, for the ASE network, familiarity and predictability are likely to be more influential in achieving successful interaction with the ASE network than the dialog initiative of the network. However, the tangibility of certain parameters is not negotiable. One such parameter is multi-threading, which refers to the multi-windowing (multi-tasking) capabilities of the network. In traditional learning design exercises, teachers are comfortable with the notion of adding events to (or deleting events from) their design on the fly (not necessarily sequentially), jumping between different events of the design until a complete whole is constructed. Inclusion of multi-threading increases the degree to which the actor can relate his interaction with the ASE environment to the traditional methods of creating a learning design.

The formalisation of the LUs, topics, events and navigational constructs enables the ASE network to have a high degree of conformity between different events and navigational constructs. On the same basis, the formal articulation of nodes enhances the network’s general predictability and generalizability, reducing the steepness of the learning curve for the actor.

The dialogue initiative of the network is shared (not necessarily equally) by the network and its actors. The allocation of pre-emptiveness depends on the specific external display of the ASE network as determined by the customisability of the network. The formal articulation of the network elements as defined above, makes it possible to design a fine-grained model for customisability. As the interaction competence level of the actor with the ASE network increases, the system pre-emptiveness gives way to actor pre-emptiveness.

5.2 Implementation of individual usability parameters on ASE nodes

Menu selections or buttons on the external display state enable the ASE network actor to define (declare) LUs and their associated topics during the design of a learning environment. Once these are defined, the actor must design the segments and nodes of the network by identifying different events that can be used to teach the identified topics. Our focus is on how the usability of the ASE interface can be increased enabling the actor to design these events.
For the purpose of this paper, we use the question-type event to describe the usability design issues for events. In general, many of the formal attributes of the different events are not obligatory. For example, the comments and hints attributes might remain undefined for most types of questions, and therefore the internal state of the ASE network may assign default NIL values to these types of optional attributes.

However, calling the actor’s attention to ‘obligatory’ attributes might enhance the usability, especially as regards the learnability curve of the novice actors. For example, a novice actor has a basic expectation of a question environment, namely a question number, a description, the question itself, the answer and credits for the question. Let us call these ‘obligatory attributes’ our primary question attributes. To enhance learnability, the network should respond to the actor’s request to create a question node by producing a perceivable coherent interface window where the primary attributes await input. A <More Advanced..> button on the perceivable question window gives the actor access to the other attributes. The network keeps a counter determining how often the actor interacts with the <More Advanced..> button, and records the measure of interaction with this button. Should this interaction be a frequent event, the perceivable interface window is adjusted, to make the most frequently accessed (or all) attributes available on the same interface window as the primary attributes.

This approach to designing the question interface window (or any other event interface window for that matter) enhances both predictability and familiarity. On the negative side, it favours system pre-emptiveness and thus restricts flexibility to a certain extent. However, buttons such as <Auto-numbering> and <Manual-numbering> allow for task-migratability, where the actor may, for example, interfere with the network’s automatic question number scheme. Also, the aforementioned <More Advanced..> button, and the network’s automatic response to the actor’s frequent interaction with this button, increases adaptivity (customisability) and thus also the flexibility of the network.

In addition, it is further suggested that the network should consider the actor’s input expressions for consistency, and make corrective suggestions where discrepancies occur. For example, if the actor defines a total of four questions, and then at some stage deletes Question 2 but neglects to renumber the remaining question, the network should then alert the actor to the discrepancy and suggest automatic correction, or allow the actor to perform a manual correction.

5.3 Implementation Example
To implement our research model we developed an ASE prototype of limited scope. The
objectives of this prototype were twofold, namely (1) to provide a usable authoring tool to develop an IC network; and (2) to provide an authoring tool that creates LOs that are based on open standards in order to foster large-scale re-use. We limited the scope of our prototype to prove that the suggested objectives are indeed achievable within an ASE network. In the rest of this section we describe the scope, architecture and implementation details of the prototype.

For the design, we limited the scope of our design to cover only the course information subnetwork of the ASE network. The objective of the course information subnetwork is depicted as general course information such as course objectives, prerequisites, syllabus, assignment outlines, et cetera.

Following our suggested waterfall approach to ASE design, we started our process by defining the LUs of the ASE. In the specific prototype, we defined only one LU, namely $L_1 = \text{COURSE INFORMATION}$. It might be useful to comment on LU as well as topic selection at this stage. In general, topic selection cannot be content-specific, and it should not be possible to define specific topics at this stage of the ASE design. In fact, specific topic identification is left to the ASE network actor while the ASE interface should merely make provision for topic identification. However, this specific LU that we have chosen for our prototype, namely COURSE_INFORMATION is included and is also similar in almost all courses, i.e. irrespective of the course, since it contains meta-information about the course such as course objectives, outcomes, et cetera. Therefore, we have defined very specific topics at this stage of the design, namely:

$$T_1 = \text{OBJECTIVES}$$
$$T_2 = \text{PREREQUISITES}$$
$$T_3 = \text{MATERIAL}$$
$$T_4 = \text{COMMUNICATION}$$
$$T_5 = \text{ASSESSMENT}$$
$$T_6 = \text{SYLLABUS}$$
$$T_7 = \text{HOW\_TO\_STUDY}$$
$$T_8 = \text{INTERNET\_ACCESS}$$
$$T_9 = \text{ASSIGNMENTS}$$

In the next phase of our design approach we defined the type of events to be made available to achieve our set objectives. Due to the limited scope assumed in the beginning, we restricted ourselves to the following events:

$$\text{EventType1} = \text{INFO}$$
EventType2 = EXERCISE
EventType3 = QUEST
EventType4 = REF

Considering each topic at a time, the following specific events were articulated. (For each of the definitions below, the attributes in small caps are input expressions that can be entered by the actor during definition of specific events.)

\[ E_{OBJ} = \{ \text{INFO}', \text{C\_OBJ'}, \text{objDescription} \} \cup \{ I_j \}, \]
where \( I = \{ \text{rangeNo}, \text{itemNo}, \text{itemType}, \text{itemDescription} \} \)

The objective event allows for a general description augmented by several specific objectives.

\[ E_{PREREQ} = \{ \text{INFO}', \text{C\_PREREQ'}, \text{prereqDescription} \} \cup \{ I_j \}, \]
where \( I = \{ \text{rangeNo}, \text{itemNo}, \text{itemType}, \text{itemDescription} \} \)

The prerequisites event also allows for a general description augmented by several specific prerequisites. (Specific prerequisites may, for example, include prior lower level prerequisite courses, hardware, access to a computer, et cetera.)

\[ E_{MAT} = \{ \text{INFO}', \text{MATERIAL'}, \text{description} \} \cup \{ \text{books} \} \cup \{ \text{guides} \} \cup \{ \text{audio} \} \cup \{ \text{video} \} \cup \{ \text{software} \} \cup \{ \text{articles} \} \]
The materials event allows for the definition of all the materials that are needed for a specific course. It is possible to define specific types of material items such as books, audio, software etcetera. Each of these specific attribute items requires specific input expressions from the actor relating to the suggested name of the attribute item. The reason for distinguishing between different types of items is to enhance search and retrieval capabilities.

\[ E_{COMM} = \{ \text{INFO}', \text{COMM'}, \text{description} \} \cup E_{URI} \]
The communication event allows for the description of the communication mechanisms of the course. This typically includes a general description with navigational constructs to external links such as an e-mail address or discussion forums.

The Internet-access event has a similar structure as the communications event allowing for a description of possible Internet sources combined with a set of Internet reference links.
The assessment event allows for elaboration on the methods of assessment of the course.

The syllabus event allows for elaboration on the course’s syllabus details. A typical syllabus event could consist of a long description field (blob type) or a general description with several syllabus items.

The **how-to-study** event has a similar structure to the *syllabus* event, allowing for elaboration on possible study methods for the particular course.

According to the given articulation of the *assignments* event, the event is composed of *information transfer-type* and *question-type* events. Selection of the *assignments* event activates a first interface window awaiting input expressions with regard to the following attributes: assignment environment, credits for the assignment, and due date by which the assignment has to be submitted for evaluation. Two input buttons are then made available to define questions within each assignment. Each of these input buttons triggers an overlap interface window requiring input expressions relating to other ‘multiple-choice’ or ‘other’ questions, depending on the selected button.

We conclude our discussion on the ASE prototype by briefly considering the events of the export transaction, in which the input expressions are mapped onto output expressions, and wrapped as XML LOs available for reuse. The export transaction interacts with an XML repository containing one-to-one mappings between the components and their corresponding XML tags. Each component has at least two tags associated with it, namely a startup-tag and a stop-tag. Furthermore, each component can also be comprised of an unspecified number of items. During the export transaction, each component iterates through the following method:

- Finding its own startup-tag from the repository and pre-affixing the tag to itself.
- Placing its associated stop-tag on top of the stack.
- Scanning for any items within it. For each nested item it finds, the procedure of pre-affixing the startup tag to the item and placing the associated stop-tag on top of the stack, is repeated.
• When a new (non-nested) item is encountered, the stop-tag on top of the stack is removed and appended to the last item. The scanning procedure is then repeated.
• When no more items for a specific component are available, the stack is emptied from top to bottom, appending each of the stop-tags still on the tag to the (now) XML-object.

Figure 5: Screen shot from ASENI prototype in the assignments environment.
Figure 5 shows a screen shot of an assignment event within the ASE prototype.

6 Conclusion
Because an ASE network forms an intrinsic subset of a set of e-learning networks, existing ASE software requires expert users who are not challenged by their computing design environments, and are therefore able to focus their full attention on the design of a learning environment. However, professors are increasingly required to design e-learning environments, and as a result are challenged by the fact that their bags of professional skills do not, by default, include computing skills and natural software usage intuition. The implication of this challenge is that
professors struggle to focus their attention on learning environment design, and are impeded by the design environment.

In this paper we proposed a cyclic waterfall approach to design an ASE network that adheres to several of the important usability parameters known in software development, and at the same time produces reusable, XML-wrapped output LOs. Our suggested approach of articulating required ASE elements as network nodes, enables the designers to separate learning design issues from interface usability issues.

We concluded our approach by describing an implementation of an ASE prototype with limited functionality. The design of the prototype relied on the suggested methodology, where different events to be included in the ASE were designed through a set of formally defined nodes. Our prototype proved to overcome many of the challenges that confront professors when they are designing e-learning software. The prototype is largely based on system pre-emptive dialog initiative, which impedes flexibility to a certain extent. However, in the domain of authoring e-learning software, the most important usability design criteria are focussing on increasing usability for novice uses rather than expert users. A full implementation would obviously also consider the expert user, and include a more user pre-emptive approach. On the positive side, our prototype greatly enhanced learnability, robustness and most flexibility parameters.

Bibliography