Using ERMIA for the Evaluation of a Theorem Prover Interface

Mike Jackson[*], David Benyon[*] and Helen Lowe[**]

[*]Napier University, Edinburgh
[**]Glasgow Caledonian University, Glasgow

{m.jackson, d.benyon}@dcs.napier.ac.uk
H.Lowe@gcal.ac.uk

Abstract

ERMIA (Entity-Relationship Modelling of Information Artefacts) provides an extension to entity-relationship modelling techniques to provide a structural representation of the interaction between people and "information artefacts". Such a representation may then be used to compare contrasting interface designs or identify potential usability problems in an existing system. In this paper we present an application of ERMIA analysis to a version of the XBarnacle semi-automated theorem proving system that features interactive proof critics.

1. Introduction

Benyon and Green have introduced a method for understanding and describing Human-Computer Interaction known as ERMIA (Entity-Relationship Modelling of Information Artefacts (Benyon and Green, 1995; Green and Benyon, 1996; Benyon, Green and Bental, in press). ERMIA uses an extended entity-relationship modelling technique to provide a structural representation of the interaction between people and computer systems or other information artefacts. This representation can then be examined and discussed between designers in order to highlight features of the interface. The construction of the model can itself reveal insights into a proposed design and the final models used to communicate between designers or between users and designers.

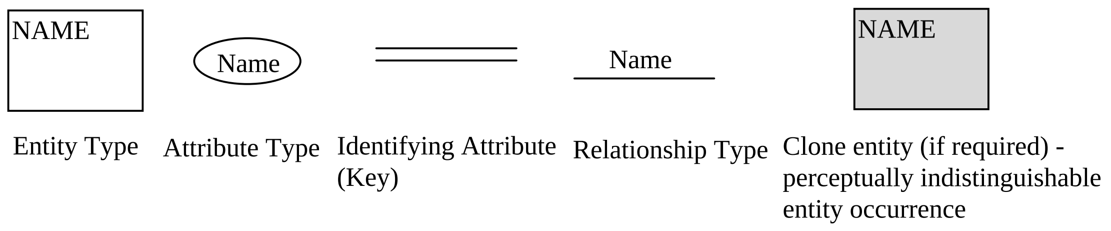
ERMIA can be used in a number of ways during interface development; to look at possible interfaces at an early stage of design, long before the final rendering has been decided on; to compare different mental models (designer's/user's, or across different users); or to analyse distributed systems, i.e. worksystems in which requisite information is distributed across different people and/or artefacts.

In this paper we show how ERMIA may be used to provide a conceptual model of a theorem prover and a perceptual model of an interface to this theorem prover. We then show how analysis of the conceptual model in itself and also with relation to the perceptual model may highlight potential usability problems. We also describe some experimental results showing how some problems identified during the ERMIA analysis then arose during an empirical evaluation of the theorem prover.

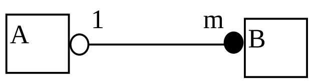
2. XBarnacle and Interactive Proof Critics

XBarnacle (Lowe and Duncan, 1997) is a version of CLaM automated proof planner (Bundy, van Harmelen, Horn and Smaill, 1990) incorporating a graphical user interface that allows users to interact with CLaM during a proof. XBarnacle is designed to allow users to step in and use their domain knowledge to guide CLaM in the search for a proof. This might be appropriate if they conclude that CLaM is pursuing an unproductive search strategy or CLaM performs a proof step the user knows is unproductive.

The version of XBarnacle described in this paper also features an implementation of interactive proof critics (Ireland, Jackson and Reid, 1997; Jackson, 1996). Proof critics (Ireland, 1992; Ireland and Bundy, 1996) provide functionality to CLaM to allow the patching of failed proof steps allowing then to succeed. Examples of proof patches include generating a required lemma, performing a case-split or revising an induction step earlier in the proof. Critics are associated with CLaM's methods and are triggered by patterns of failure of the related methods preconditions. Proof critics can extend the power of CLaM allowing it to prove theorems previously beyond its reach. Interactive proof critics allow a user to interact with a proof critic and view all the possible patches that a critic proposes and to apply, customise or reject these. Interacting with proof critics may improve the efficiency of CLaM over the purely automated critics version and also allow theorems to be proven that are beyond the reach of the automated CLaM. Part of the functionality of the interactive proof critics is an explanation facility which describes why a method failed in terms of its preconditions, why a critic was applicable, in terms of failure of the associated methods preconditions, and what the critic will do.


NAME
Name
Name
NAME
Entity Type
Attribute Type
Identifying Attribute (Key)
Relationship Type
Clone entity (if required) - perceptually indistinguishable entity occurrence

Degree of a Relationship and Participation Conditions:



Each instance of entity A may relate to one or more instances of entity B.
Each instance of entity B must relate to one instance of entity A

Figure 1: Basic Notation of Constructs in ERMIA

3. An Introduction to ERMIA

The entity-relationship (E-R) model is a graphically-based technique for representing the things of interest (entities) in an application and the associations between them (relationships). An Entity type is an aggregation of one or more property (or attribute) types. The concept of an entity provides two types of abstraction. The aggregation of properties into entities allows the designer to focus on the entities and to suppress details of the attributes. The classification of entity occurrences as entity types allows the designer to deal with a class of things rather than the individual things themselves. For example the methods (specifications of tactics) used by CLaM can usefully be viewed as instances of an entity *METHOD*, say, which has attributes *Name* and *Definition*, and so on.

Entities in the same set have the same types of attribute, though typically these attributes will take different values for different occurrences of the entity. For example, each method will have a different value for the *Name* attribute. Entities are defined by their attributes. The characteristics which define an entity are obtained by analysts in consultation with users. ERMIA does not accept that there is an objective world waiting to be carved up into a universal set of entities. Entities are subjective. Defining the entities makes such subjectivity explicit.

A further level of abstraction may be obtained by recognising that entities can have sub-types. This allows us to generalise certain characteristics or relationships between entity super-types, whilst recognising that the sub-types may differ from the super-type in some (relatively) minor respect. For example as XBarnacle allows user-CLaM collaboration during a proof we have the notion of an *AGENT* entity with sub-types *USER* and *CLAM* (the CLaM planner) as both these entities may take actions in CLaM. Each sub-type of an entity may share some attributes and/or relationships with their super-type entity but differ in others. Entities in ERMIA also demonstrate the principal of encapsulation. It is possible and often desirable to deal with quite complex artefacts as if they were a single entity, hiding the details of their construction. This type of abstraction again delivers a degree of simplification which makes for a more powerful model.

Conceptual entities, or concepts, are cognitive constructs. Conceptual entities can be seen as having some correspondence with the ideas or notions which users and/or designers have in their minds. We develop concepts in order to make sense of the experienced world. We represent those concepts and the relationships between them by developing ERMIA models. Perceptual entities are things in the experienced world which are of interest to the ERMIA modeller within the terms of some discourse. They are defined at some level of abstraction which is suitable for the intended perceivers.

In ERMIA, entities (but not relationships) have attributes (also known as properties or characteristics). An entity is the aggregation of its attributes in that it is defined as the total of its attributes. Usually one or more of the attributes are used to distinguish between entity occurrences. This attribute (or attributes) is known as the entity *identifier*. For example *Name* may be considered to be the identifying attribute of the *METHOD* entity since each method used by CLaM has a unique name. The structural attributes of perceptual entities are their perceivable characteristics (typically visual, audible or tactile properties). Behavioural attributes describe perceptual changes which occur under certain circumstances. An important

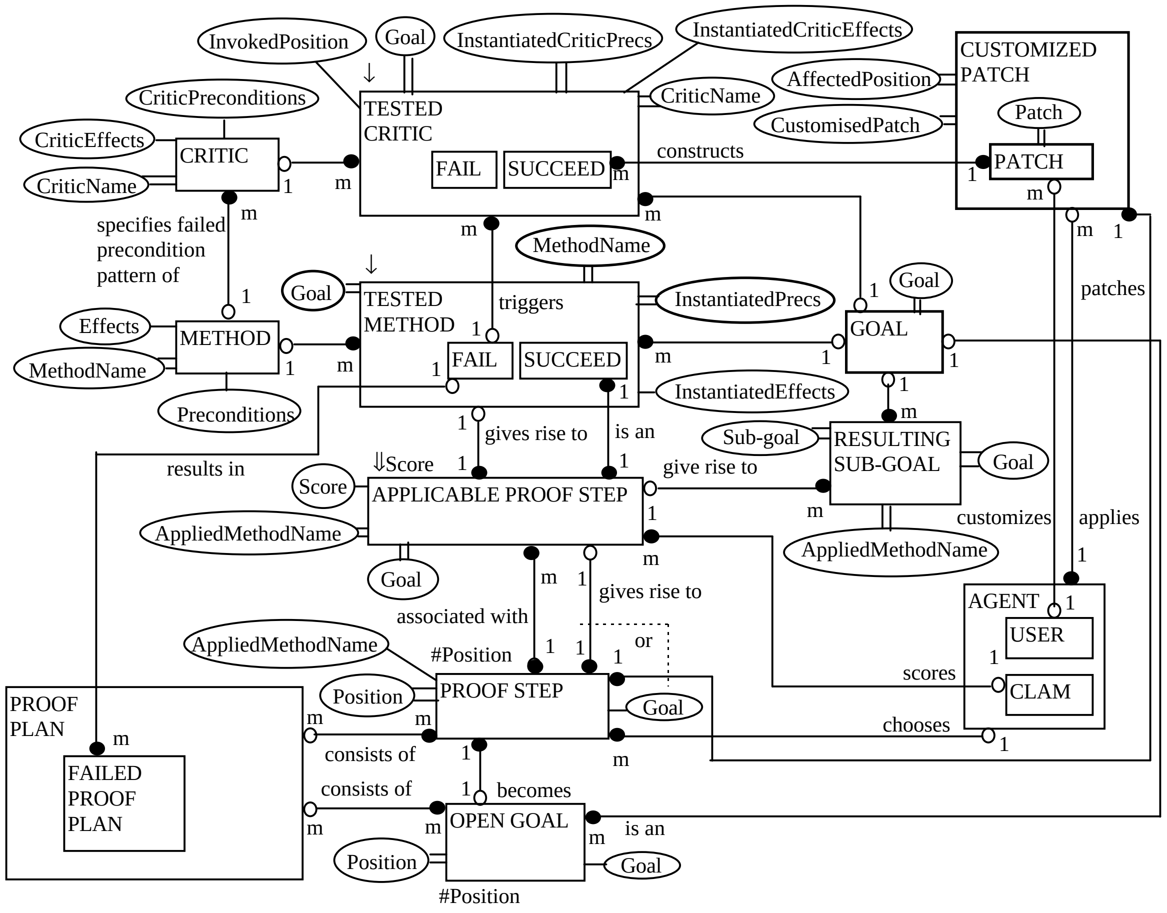


Figure 2: A Conceptual ERMIA for XBarnacle 3.2

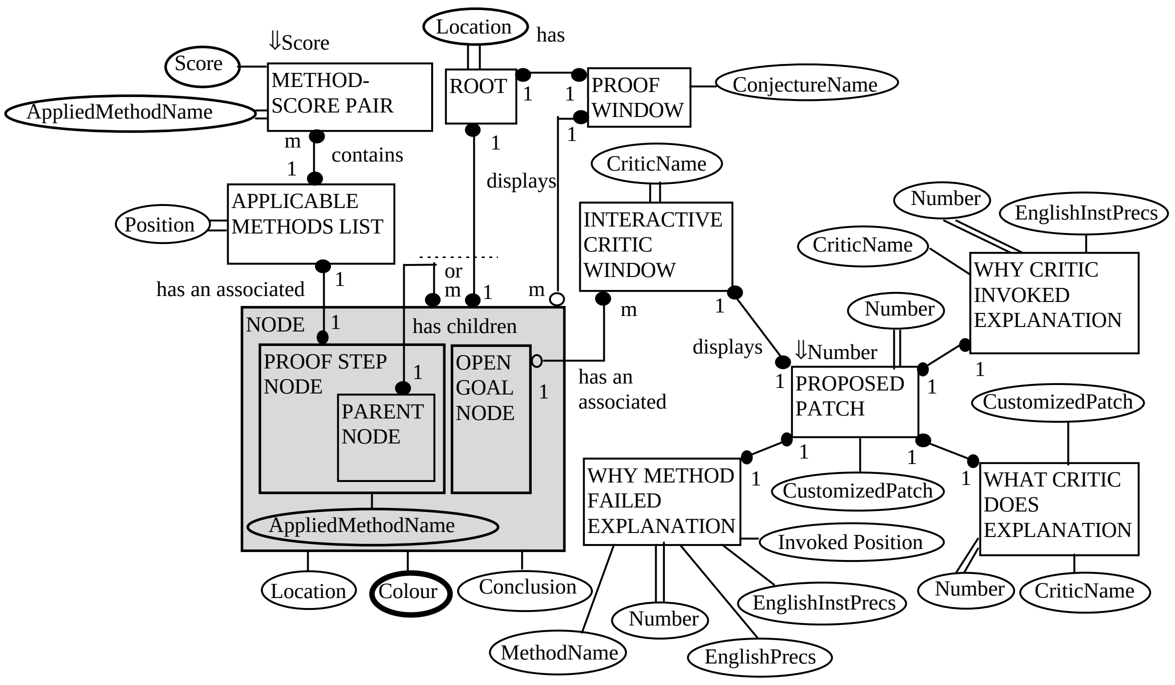
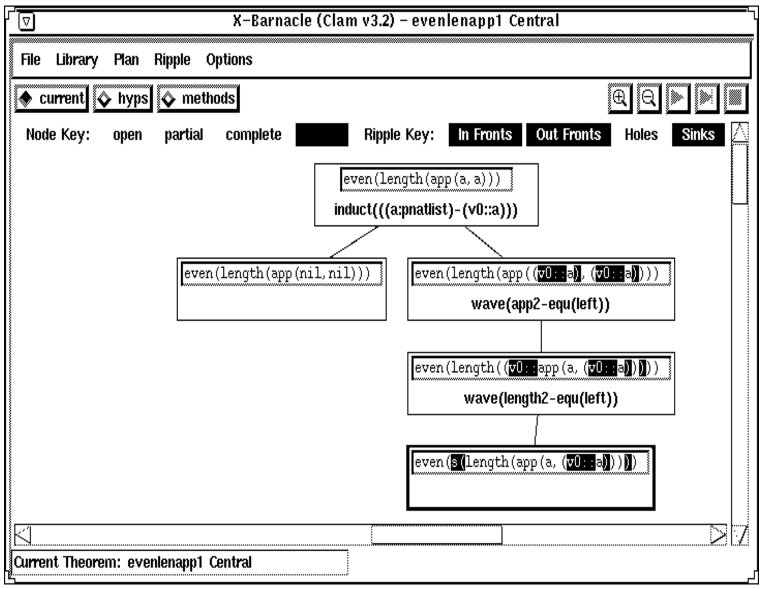
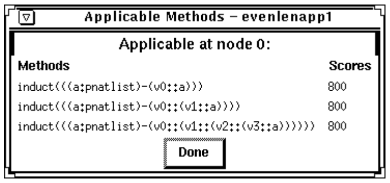


Figure 3: A Perceptual ERMIA for the XBarnacle 3.2 Interface



```
even(length(app(a,a)))
induct(((a:pnatlist)-(v0::a)))

even(length(app(nil,nil)))

even(length(app((v0::a),(v0::a))))
wave(app2-equ(left))

even(length((v0::app(a,(v0::a)))))
wave(length2-equ(left))

even(s(length(app(a,(v0::a)))))
```

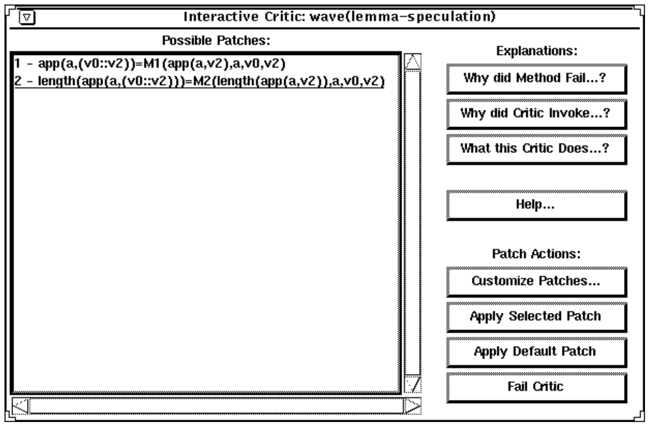
The main XBarnacle interface



Applicable at node 0:

Methods	Scores
induct(((a:pnatlist)-(v0::a)))	800
induct(((a:pnatlist)-(v0::(v1::a))))	800
induct(((a:pnatlist)-(v0::(v1::(v2::(v3::a))))))	800

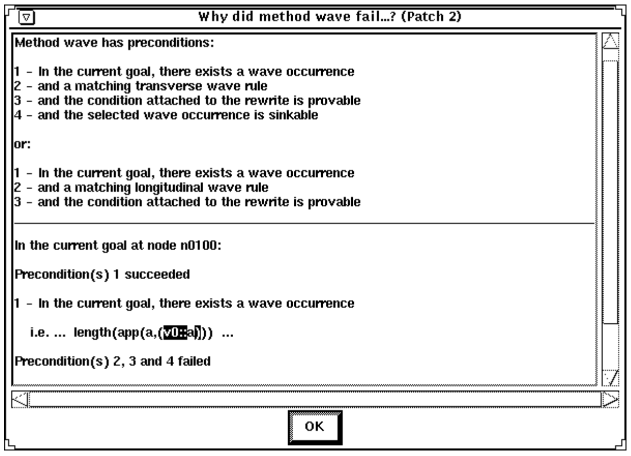
The list of methods (and heuristic applicability scores) that the user may obtain from a node



Possible Patches:

1 – app(a,(v0::v2))=M1(app(a,v2),a,v0,v2)
2 – length(app(a,(v0::v2)))=M2(length(app(a,v2)),a,v0,v2)

The interactive proof critic window which XBarnacle displays when a critic is invoked (the main part of the window displays the proposed proof patches)



Method wave has preconditions:

1 – In the current goal, there exists a wave occurrence
2 – and a matching transverse wave rule
3 – and the condition attached to the rewrite is provable
4 – and the selected wave occurrence is sinkable

or:

1 – In the current goal, there exists a wave occurrence
2 – and a matching longitudinal wave rule
3 – and the condition attached to the rewrite is provable

In the current goal at node n0100:

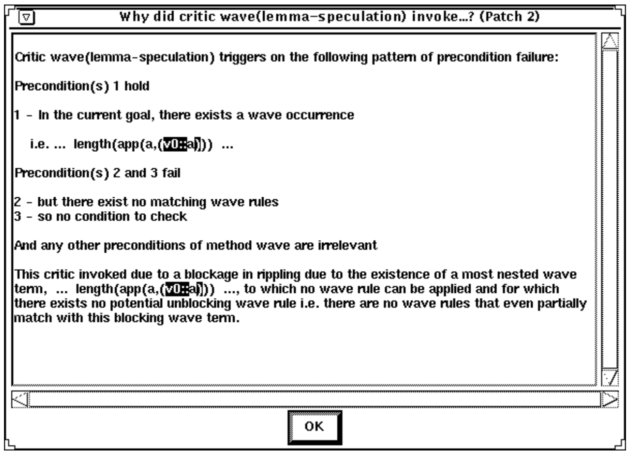
Precondition(s) 1 succeeded

1 – In the current goal, there exists a wave occurrence

i.e. ... length(app(a,(v0::a))) ...

Precondition(s) 2, 3 and 4 failed

An explanation from the interactive proof critic as to why the method failed (acquired by pressing the appropriate button)



Critic wave(lemma–speculation) triggers on the following pattern of precondition failure:

Precondition(s) 1 hold

1 – In the current goal, there exists a wave occurrence

i.e. ... length(app(a,(v0::a))) ...

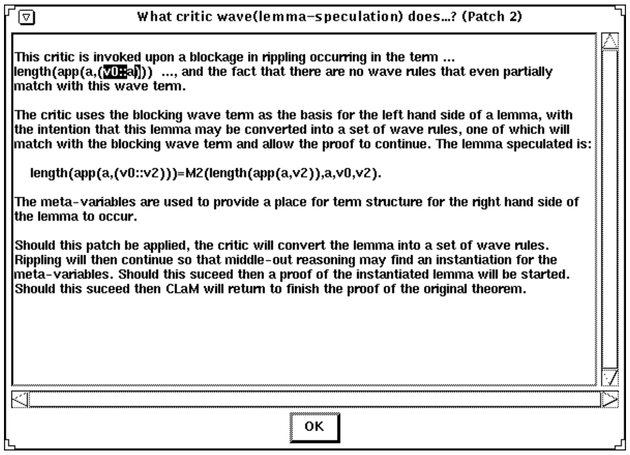
Precondition(s) 2 and 3 fail

2 – but there exist no matching wave rules
3 – so no condition to check

And any other preconditions of method wave are irrelevant

This critic invoked due to a blockage in rippling due to the existence of a most nested wave term, ... length(app(a,(v0::a))) ..., to which no wave rule can be applied and for which there exists no potential unblocking wave rule i.e. there are no wave rules that even partially match with this blocking wave term.

An explanation from the interactive proof critic as to why the critic invoked (acquired by pressing the appropriate button)



This critic is invoked upon a blockage in rippling occurring in the term ... length(app(a,(v0::a))) ..., and the fact that there are no wave rules that even partially match with this wave term.

The critic uses the blocking wave term as the basis for the left hand side of a lemma, with the intention that this lemma may be converted into a set of wave rules, one of which will match with the blocking wave term and allow the proof to continue. The lemma speculated is:

length(app(a,(v0::v2)))=M2(length(app(a,v2)),a,v0,v2).

The meta-variables are used to provide a place for term structure for the right hand side of the lemma to occur.

Should this patch be applied, the critic will convert the lemma into a set of wave rules. Rippling will then continue so that middle-out reasoning may find an instantiation for the meta-variables. Should this suceed then a proof of the instantiated lemma will be started. Should this suceed then CLaM will return to finish the proof of the original theorem.

An explanation from the interactive proof critic as to what the critic will do (acquired by pressing the appropriate button)

Figure 4: Various components of the XBarnacle interface

development for ERMIA models is that perceptual entities are not always distinguishable from one another. Thus we introduce the notion of a ‘clone’; an entity type which has instances which are perceptually indistinguishable from other instances of that entity.

In ERMIA, as in ER models, entities are associated with each other and sometimes with themselves through relationships. There may be more than one relationship between entities. A one-to-one relationship (*1-1*) between entities *A* and *B* associates an occurrence of entity *A* with at most one occurrence of entity *B* and an occurrence of entity *B* with at most one occurrence of entity *A*. A one-to-many relationship (*1-m*) between entities *A* and *B* may associate many occurrences of entity *B* with each occurrence of entity *A*, but each occurrence of *B* is associated with at most one occurrence of *A*. A many-to-many relationship (*m-m*) permits many occurrences of entity *B* to be associated with each occurrence of entity *A* and many occurrences of entity *A* to be associated with each occurrence of entity *B*. It is useful to decompose *m-m* relationships by the introduction of a new entity. For example there is a potential *m-m* relation between goals and methods since a method may be applied to a number of goals and each goal may have a number of methods applicable to it. In Figure 2 we have broken up this *m-m* relationship revealing the entity *TESTED METHOD*, resulting from the application of a specific method to a specific goal.

Further semantics of relationships are represented by including participation conditions of entities in relationships. Mandatory participation constrains the entities in a set so that they must always participate in the relationship. Optional participation allows some or all occurrences of an entity not to participate in the relationship at any particular time. Sometimes it is desirable to insist that an entity must participate in two or more relationships (inclusivity). This is represented on an ERMIA diagram by suitable annotation of the diagram. Similarly we may want to represent that an entity may only participate in one of several relationships (exclusivity). Other constraints on the participation of entities in relationships may be represented by natural language annotations.

The basic notation used for ERMIA is shown in Figure 1. Figure 2 presents an ERMIA of the conceptual elements in XBarnacle.

4. A Perceptual ERMIA of the XBarnacle interface

The XBarnacle interface may be viewed as a viewport onto the underlying conceptual domain. In Figure 3 we present a perceptual ERMIA of this viewport, components of which are shown in Figure 4. Where conceptual entities and attributes are rendered at the interface we have used the same entity and attribute names as in the conceptual model of the underlying CLaM system. Note that there are new entities, however, for example *METHOD-SCORE PAIR* or *WHY METHOD FAILED EXPLANATION*, which have no specific conceptual analogue.

Note that nodes (denoting a super-type of the perceptual entities representing *PROOF STEPS* and *OPEN GOALS*) have a perceptual attribute, *Colour*, and that the value of this attribute directly reflects the type of node (i.e. is it a proof step node or an open goal node). Note also that nodes in the proof plan as displayed by XBarnacle are clones as there may be no way to tell certain occurrences of nodes apart at the interface. This may have serious implications for the user as we describe in the next section.

5. Using ERMIA to Identify Potential Usability Problems

We now give examples of how analysing the conceptual ERMIA in itself, and also comparing the conceptual ERMIA to the perceptual ERMIA of the viewport, can highlight potential usability problems. The work on ERMIA models of XBarnacle was done as part of research into the utility and usability of interactive proof critics. A co-operative style evaluation (Monk, Wright, Haber and Davenport, 1993) has been performed to address this question. One of the aims of this evaluation was to see if the problems highlighted by an ERMIA analysis undertaken prior to the evaluation arose in actual use of the interface by real users, thereby giving evidence as to the utility of conducting an ERMIA analysis. When discussing the problems highlighted by ERMIA we shall give examples where those problems arose in practice.

Problem 1. A Problem Due to the Collaborative Nature of the Interface

From our knowledge of how XBarnacle is used we know that a proof step may have been chosen by the CLaM planner or the user. However our ERMIA model shows that neither the *PROOF STEP* nor *APPLICABLE PROOF STEP* entities (of Figure 2) of XBarnacle contain any attribute to record which agent actually applied each proof step. Thus the system is limited in that there is no means of determining the division of labour (if any) between the CLaM planner and a user when performing a proof. Related to this is the fact that critics may also be responsible for applying proof steps and, again, no means of storing this fact, in such cases, is provided.

This is important since users and other interested parties may over-estimate or under-estimate the power of CLaM or may gain a false impression of the reasoning strategies used by CLaM if this information is not available to them. An example of this arose during the evaluation. One participant, an expert in CLaM and proof critics, remarked on being presented with a proof:

“...so its chosen an induction on a, double induction on a which was very clever of it. How did it manage to think of a double induction? That’s cunning.”

The participant was unaware that the double induction resulted not from a method application, as they assumed, but rather from a critic which may redo induction steps. Another participant, also an expert in CLaM stated during the same example:

“That’s no normal induction analysis...that’s somebody being clever”

This problem is an example of how providing functionality at the interface (in the case of user/CLaM collaboration) or providing conceptual and/or interface functionality (in the case of proof critics or interactive proof critics) can create the need for new attributes in certain underlying conceptual entities to support the implications of this additional functionality. In this example this would perhaps entail the addition of an attribute to the conceptual *PROOF STEP* entity to identify who executed each step in the proof (or if the proof step arose from a critic application) and the provision at the interface of a suitable presentation of this new attribute.

Problem 2. Positions in the Proof Plan

Figure 3 shows how XBarnacle displays proof steps and open goals using a node entity. Analysing the ERMIA we see that this entity (and hence its rendition at the interface) has no identifying attribute meaning that nodes at the interface are clones - node entities do have an attribute *Location*, the location of the node on the XBarnacle display, but this may change as a proof progresses and is unrelated to the underlying position of a proof step or open goal in the proof plan. This demonstrates a problem with the interface since the proof steps and open goals in the underlying theorem prover, which nodes at the interface represent, *do* have an identifying attribute - their position in the proof plan, as may be seen in Figure 2. Therefore the interface may, in certain circumstances, cause navigation problems for the user if two separate parts of a proof plan have the same sets of proof steps or open goals as these will be indistinguishable at the interface. Also, referring to proof steps or open goals in the proof plan by position may cause problems since there is no direct representation of this position in the entities that display the proof plan - the user must take extra action to display the position of a node in the proof plan.

A problem of this type arose in the evaluation. For example the induction revision critic which may propose the revision of an application of the induction method at a proof step earlier in the proof plan prints as patches to the user information of form:

```
Apply method induct(x:pnat,s(x)) at node 000
```

where *000* is a proof step/node position in the conceptual proof plan. One participant in the evaluation pointed at the displayed proof plan and remarked:

“I think you need to label these nodes if you're going to refer to them by some number ... its not obvious which one you're talking about.”

despite these addresses being in a form similar to that in which node addresses are usually presented (as another participant correctly identified). Another participant stated on the same task:

“...so the question is where's node 000 ?”

and like the first participant had to head to the root of the proof plan and count down to the correct point in the proof, which would be very problematic in large proofs, as one participant stated. Another participant stated:

“I want it to do the induction that its suggesting but I want to do it on this node.”

pointing to the node where the induction *would* be done and assuming wrongly that it gets done at the current node, where the critic was invoked. The participant here did not pick up the fact that *000* referred to the node at which the induction would be done.

Unlike Problem 1 this problem arises as a result of a key attribute of important theorem proving entity (proof steps and open nodes) not being rendered directly at the interface.

Problem 3. Where was the critic invoked and to what does it apply ?

In Figure 2 we see, by following the appropriate relations and examining the attributes, that critics are invoked at specific positions in the proof plan, those positions corresponding to the position of the open goal where the associated method fails. However, we also see that the effect of a critic may be to take action at a different node in a proof tree, for example the induction revision critic described above. Related to Problem 2 problems relating to positions of proof steps and open goals in the proof plan may arise due to the interface not rendering these attributes at the interface (as is highlighted by the omission of such attributes from the perceptual ERMIA of Figure 3). Firstly the interactive critics interface, when invoked by CLaM, does not display the node at which it is invoked as one participant stated:

"Which goal's it working on now...which one's it asking about ?"

The user must take extra action to elicit this information, as one participant verbalised:

"I'm hitting the "Why did method fail ?" button which tells me which node the problems at which isn't entirely clear unless you actually do something like this..."

causing the explanation as to why a method failed to be displayed, this explanation (as Figure 3 shows) rendering the conceptual attribute *InvokedPosition*, which stores the position in the proof plan of the goal to which the critic was invoked from. The participant later stated:

"...it really would be useful if the display, the interactive critic window tells you which node its looking at..."

Similarly the critic interface does not always say to which node it does apply. Nor do the explanations. This led to comments of the form:

"I think it would be quite useful if the...the display actually showed which nodes they were proposing to be applied to without actually having to hit one of the buttons to get the more detail."

and

"It certainly would be useful if you could see what nodes each of the patches were applying to.".

The utility of showing the nodes to which a critic is applied was borne out by a comment from a participant with respect to the display of patches for induction revision which do state the node they affect:

"It's more clear from these what they're going to do which is apply an induction at a particular node."

Again these problem arises from the interface not rendering certain attributes of conceptual entities in the appropriate place i.e. here the position where the critic invoked and the position to which each of the proof patches apply should be rendered in the main interactive critic window, not just in the explanation windows which pop-up only after extra action by the user.

Other Problems

(Jackson, Benyon and Lowe, 97) describes in detail other potential usability problems that may arise, most of these also relate to the standard XBarnacle system described in (Lowe and Duncan, 97). The problems include:

- As stated proof critics may create a lemma automatically. This sets up a requirement for the lemma to be proven. CLaM has functionality to prove such lemmas automatically resulting in a system where a conjecture may either have been defined by the user or a critic. This leads to a problem related to Problem 1 in that false impressions of XBarnacle's power may arise if outside observers are unaware of this fact. Therefore some means of recording who defined what conjecture should perhaps be provided.
- Each applicable proof step has an associated set of resulting sub-goals but only the sub-goals for the applicable proof step actually applied may be accessed (since these become sub-goals in the proof plan);

- Method applicability is determined by their preconditions but there is no way of accessing the preconditions of a method as they relate to a goal in the proof plan i.e. one cannot see why a method was applicable to a given goal. Nor can one see why other methods failed to be applicable to a goal i.e. the pattern of precondition failure. This is important since a failed method may lead to a failed proof plan. The exception is for methods whose critics invoke, the precondition pattern may then be viewed using the interactive critic interface. One participant in the evaluation used this feature extensively and this may give indications as to the utility of this form of explanation in general.

6. Conclusion

We have presented an introduction to ERMIA and a model of the conceptual structure of a version of XBarnacle that features interactive proof critics. We also provided a perceptual model of a viewport onto that conceptual structure and showed how analysis of the conceptual structure, both in itself and in relation to the perceptual structure, highlighted potential usability problems, some of which arose when potential users of XBarnacle participated in an evaluation of the utility and usability of interactive proof critics.

There is little doubt that developing the ERMIAs has provided an insight into XBarnacle. Whether such insight could have been gleaned through other approaches is a moot point We would argue that a task analysis approach would not have highlighted some of the usability problems, because we are not dealing with existing tasks, rather we are dealing with the distribution of knowledge throughout the underlying system and the representation of this knowledge at the user interface.

This is not however to state that task analysis approaches or other interface modelling techniques are of no use. On the contrary in many respects these approaches may be superior to ERMIA. For example one limitation of ERMIA is the problem of highlighting the fact that some entities may exist only for a certain limited period of time and then cease to exist in a conceptual system. This further serves to emphasise the fact that ERMIA is one of a number of modelling techniques of great use in interface design and that interface designers may need to consider the pro’s and con’s of each of techniques, in conjunction with their own areas of concern, to choose the tools most suitable for their task.

References

Benyon, D. R. (1996) Domain Models for User Interface Design. In Benyon, D. R. and Palanque, P. *Critical Issues in User Interface Systems Engineering*, Springer-Verlag

Benyon, D. R. and Green, T. R. G. (1995) Displays as Data Structures. In Nordby, K., Helmersen, P. H., Gilmore, D. J, and Arnesen, S. A. (Eds.) *Human-Computer Interaction: INTERACT-95*. London: Chapman and Hall.

Benyon, D. R. and Green, T. R. G. and Bental, D. (in press)*Conceptual Modelling for User Interface Design, using ERMIA* Springer

Bundy, A. Van Harmelen, F. Horn, C. and Smaill, A. (1990) The Oyster-CLaM System *10th International Conference on Automated Deduction* Kaiserlauten, Germany July 1990 (ed. M.E. Stickel) Springer-Verlag: London, Lecture Notes in Artificial Intelligence-449 p647-648

Green, T. R. G. and Benyon, D. R. (1996) The skull beneath the skin; Entity-relationship modelling of Information Artefacts. *International Journal of Human-Computer Studies* **44**(6) 801-828

Ireland, A. and Bundy, A. (1996) Productive Use of Failure in Inductive Proof Special edition of *Journal of Automated Reasoning* on Inductive Proof **16** (March 1996) 1996.

Ireland, A. (1992) The Use of Planning Critics in Mechanizing Inductive Proofs In *International Conference on Logic Programming and Automated Reasoning - LPAR'92* St. Petersburg July 1992 (ed. A. Voronkov) Springer-Verlag: London, Lecture Notes in Artificial Intelligence-624 p178-189

Ireland, A., Jackson, M. and Reid, G. (1997) A collaborative approach to theorem *proving Proceedings of the First International Workshop on Proof Transformation and Presentation* Schloss Dagstuhl, Germany, April 1997 (eds. X. Huang, J. Pelletier, F. Pfenning and J. Siekmann) p21-22

Jackson, M. (1996) *HCI Techniques for Theorem Proving* MSc Project Report Heriot-Watt University, Edinburgh June

Jackson, M., Benyon, D. And Lowe, H. (1997) *Evaluating an Interface to a Theorem Prover: An Application of Entity-Relationship Modelling of Information Artefacts*. Submitted to BCS-HCI 1998 Sheffield.

Lowe, H. and Duncan, D. (1997) XBarnacle: Making Theorem Provers More Accessible, *Proceedings of the Fourteenth Conference on Automated Deduction (CADE-14)* Townsville, Australia, July 1997 (ed. W. McCune) Springer-Verlag: London, Lecture Notes in Artificial Intelligence-1249 p404-407

Monk, A., Wright, P., Haber, J. and Davenport, L. (1993) *Improving Your Human-Computer Interface: A Practical Technique* Prentice-Hall International: New York.