Accommodating Diverse Users in ETAG and ETAG-Based Design: task knowledge and presentation

Geert de Haan

Maastricht McLuhan Institute
Maastricht University
P.O. Box 616
6200 MD Maastricht
The Netherlands
g.dehaan@mmi.unimaas.nl

Abstract
This paper discusses ETAG, a formal model for design representation and ETAG-based design, a method for user interface design. The paper discusses and exemplifies the two main facilities in ETAG to describe the differences and similarities in task-knowledge requirements between users. Using evidence from the Comris project (van der Velde, 1997), conclusions are drawn about ETAG-based design as a single-notation design method, and about the tension between ETAG as a model for design representation and as a representation of user competence knowledge.

1. Extended Task-Action Grammar

ETAG (Extended Task-Action Grammar; Tauber, 1988, 1990; de Haan, 2000a) is a formal language to represent user interfaces in terms of the knowledge that a perfectly knowing user would have (in a mental model) about performing tasks. To create a psychologically valid description of user interface for design purposes, ETAG stratifies user interface knowledge into a number of levels using existential logic and written down in a formal grammar. Stratification into levels intends to meet the existence of levels in human knowledge and to reflect the major decisions that occur during the design process. An ETAG representation consists of a canonical basis, a user virtual machine, a dictionary of basic tasks, and a section with production rules.

Comris (van der Velde, 1997) is an agent-based system to provide visitors of large conferences with context-sensitive information about nearby events and people, based on a personal interest profile, an active badge system, and information about the conference schedule. In the Comris system, users carry a wearable computer, which stores a list of advisory messages from the agent system for textual and/or spoken presentation. By means of the wearable, users may, among others, play, present, delete and respond to messages, move to a previous or a next message, and adjust the volume and the minimal interest threshold.

The canonical basis of the ETAG representation of Comris contains general concepts such as object, position, state, attribute and event, to define the specific concepts that the system uses. The type hierarchy included the concepts that the user comes into contact with, such as messages and message list, and the concepts that the user may influence, such as the interest threshold that messages should exceed to be included in the message list. In addition, the type hierarchy specifies an agent concept as the external source of new messages, as well as interest and location concepts which are necessary to understand how the user's interests and whereabouts influence the advisory information in newly created messages. In the type specification each of the concepts in the type hierarchy is defined, and additional concepts and attributes, such as current_message and message_position are added as 'semantic sugar' to describe how the system works, as experienced by the user. Part of the type specification of Comris' UVM will read as follows:

```
type [message_list isa object]
  themes: message
  places: place_pos(x)(message_list)
  attributes: max_pos, current_pos

  type [message isa object]
    attributes: message_text, next, previous
```
The specification states that a message_list provides space for messages, which each take up one place in the list, and that messages have a certain text, and a previous and next attribute which will be used later on, to describe the task of selecting another message to take the current_position.

The event specification is the part of the UVM describing the workings of the system as it virtually appears to work from the point of view of the users, which may be different from how it is actually build to work. Part of Comris' event specification will read as follows:

```
@type [create_message isa event]
parameters: relevance, threshold, message_list, message_text
precondition: relevance > threshold
  create_on(message_list, max_pos, message)
  copy_to(text, message)
  current_pos = max_pos
  max_pos += 1
  play_message

The event specification describes how the system processes tasks from the user and other agents using a pseudo computer program notation. The create_message event reads as follows: when the relevance of the new message surpasses the user's interest threshold, it is added at the end of the user's message list. It then becomes the current message and an event is raised to play it (present it as speech and text).

The dictionary of basic tasks lists the tasks that the system provides to the user, and it links the command sequence, as described in the production rules, to the event specification. In the original proposal of ETAG (Tauber, 1988, 1990) it excluded system tasks and tasks which require user decisions during execution. Embedded tasks were introduced to describe system functions like those with clear visible effects, and so-called menu tasks were introduced to describe higher-level user tasks, such as making multiple selections from a list. The following task illustrates the dictionary of basic tasks:

```
@type [create_message isa embedded task]
  effect: [create_message isa event][play_message isa task]
  remark: a system task to present a new message to the user
```

To complete the ETAG specification of the Comris system, the perceptual interface should be specified next. This consists of the user perceiveable aspects of the system. It is not (yet) possible to specify the presentation interface in ETAG which is concerned with the visual design. It is better to specify the user's knowledge about the visual parts of a user interface pictorially, and for design purposes, it is easier to use so-called interactive user interface builders. The specification of the production rules which describe the command procedures will be illustrated in section about design representation for diverse users/user task world presentation.

2. ETAG-Based User Interface Design

In ETAG-Based Design (de Haan; 1996; 2000a) user interface design is regarded as the incremental specification of the mental model of a perfectly knowing user. The design process is structured into a number of discrete steps, each covering a specific set of design decisions: task- and context analysis, task design or task synthesis, conceptual user interface design, and perceptual user interface design, which covers the presentation interface and the interaction language. ETAG-Based Design was inspired by the notion that user interface design is not so much concerned with creating software but rather with providing a particular view on a task domain. Given that the user's view of the task domain (the user's task world) and the system's view should correspond for optimal task performance, and that the two views are mediated by the knowledge that users (should) have about performing tasks, it seems reasonable to develop a method which represents user interface designs by means of modelling required user knowledge.

An important characteristic of ETAG-based design in comparison to other design methods, such as MUSE (Lim and Long, 1994), Paternò's Model-based approach (Paternò, 1999), or Usage Centered Design (Constantine and Lockwood, 1999), is that, as far as possible, it uses a single notation throughout the design process. In ETAG-Based Design the ETAG notation is used to represent the analysis and design results. For this, it is necessary that the notation is flexibly adapted, beyond its original definition, to meet the specific purposes of all design stages. Originally, ETAG was intended only for user interface specification and not for representing the
results of task analysis and task design. By altering the level of abstraction of the specification, the amount of detail, and the inclusion of special modelling concepts, the ETAG notation becomes useful for different purposes. For example, in modelling business procedures during task analysis, the representation is specified at a high level of abstraction without much detail, and special concepts are used to represent agency and ownership. In contrast, when the representation of the user interface is intended for conceptual design or for prototype generation, the representation includes more details at lower levels of abstraction and excludes special concepts.

Using a single notation throughout design facilitates the transitions from one design stage to the next. It also makes it easier to create tool support, which is particularly relevant for presenting the ETAG representation as a formal representation in a suitable way to all the stakeholders. A final advantage of using a single notation is that designers themselves are not required to learn and use a variety of different notations. In the approaches of Paternò (1999) and Constantine and Lockwood (1999), for example, designers must be able to deal with a variety of notations, in contrast to only one in ETAG-based design.

3. Design Representation for Diverse Users

In ETAG-based design, differences between groups of users can be accommodated in two principally different ways, depending on whether users share common knowledge about the user’s task world, and whether they share a common presentation of the user task world.

3.1 User task world knowledge

During task design, when the results of task analysis are syntheised into new task specifications, it may become clear that the design involves multiple users with rather different roles, tasks, and privileges in the work situation. In this case, different users have different knowledge requirements with respect to performing tasks and, consequently, it is necessary to create multiple ETAG representations of the UVM and the dictionary of basic tasks. Examples are, software to support business processes in which users have different responsibilities, and adaptive systems, which, provide different task sets, depending on user characteristics, such as experience (Oppermann, 1994). In the Comris example, conference administrators and ordinary users play a different role. Users and administrators use the same browser interface to provide the system with information and instructions, but administrators are also able to chance the conference schedule or to create and distribute messages. In effect, the task dictionaries are very similar but the UVM and the type specifications are considerably different.

The need to create more then version of the higher levels of the ETAG representation does not imply that separate design tracks are required or that the representations will be distinctly different. First, regardless of how many different types of users may be distinguished, they share the same work situation and need to have more knowledge in common than in separation. Secondly, at some point in the design process it may be practical to join the different representations, for example, to specify the implementation of the software, but, in principle, when users require different mental models, also different high-level ETAG representations are required.

3.2 User task world presentation

A different situation occurs when users need the same knowledge to perform their tasks but don’t want to or cannot access the system in the same way. This situation occurs when different users or different work situations require that the user task world is presented in different ways, such as, for example, browsing the internet by means of mobile hardware, or using Braille or speech-output by blind users. In ETAG terms, the user virtual machine for browsing the web by means of a desktop PC or a web-enabled mobile phone are similar. In these cases, a single ETAG representation suffices but different presentation interfaces and production rules must be specified.

In the Comris system, users may choose between accessing advice messages by means of a small wearable, worn on the breast with a shoulder strap, and by means of an information kiosk, or rather, a concealed PC, which is also used to enter information and instructions to the agent system. With respect to browsing the user’s message list, both devices provide the same information and similar functions: to start and stop playing a message, to start and stop displaying a message, to respond with Yes or No, to select the next or the previous message, to delete a message, and to increase or decrease the interest threshold.

The Comris kiosk is supplied with a pointing device and a keyboard, whereas the wearable only has five buttons and a volume slider. The audio output is similar for both devices. For visual output, the kiosk is supplied with a PC screen, whereas the display area of the wearable device is limited to 2 lines of 16 character positions, also used to display labels for each of the buttons. Given the physical limitations of the wearable device, large part the dialogue and the presentation interface has to be laid-out in time whereas the kiosk interface can use a spatial layout.
The kiosk interface provides sufficient space to present the list of messages in textual form but the wearable device has to use either scrolling text or abbreviated versions of the messages. In a usability experiment (de Haan, 2000b) both options were used: messages were presented by scrolling text when displaying a message, and presented in abbreviated form when selecting a next or previous message. The kiosk provides sufficient space to represent each function by a separate button. On the wearable there are more functions than there are buttons available, and alternative function-to-button mappings are required by means of dialogue steps or an additional mode function.

Four different interfaces were designed for the wearable device, each according to a general principle to map between functions and buttons, such that one interface used a control key and three others used a combination of modes and dialogue steps. The presentation and the dialogue interface are very different from those of the kiosk.

In ETAG, the user-system dialogue is represented in the production rules, which specify how basic tasks are translated into the physical actions to invoke them, along 4 levels of specification which describe, respectively: the command line syntax, the way of referring to commands (pointing, naming, etc.), the labelling of command elements, and the physical actions. The specifications of the wearable and the kiosk interface have in common that a single interaction style is used (pointing) and the physical actions are virtually identical (pressing buttons). Also, each button is labelled with the name of the command and, consequently, the lexical-level of production rules is superfluous. In effect, this means that all four levels of the production rules collapse into one level, describing the relation between a particular task and the associated action.

Specifying the kiosk interface is straightforward because there is no need for dialogue steps. Since, both, the list of textual messages and the list of function buttons fit one display screen, the presentation interface may be designed as a single web page and the dialogue interface only requires a single one production rules for all basic tasks:

$$T[task] ::= \text{click\_button}[task]$$

Specifying the wearable interfaces is more complex because there is insufficient space for both messages and buttons. That messages are presented as scrolling text or in abbreviated form is relevant only to the presentation interface and not to the production rules because the presentation modes are associated with different functions. That functions are invoked by means of different dialogues is important to the design of the presentation interface because each state of the dialogue requires a description of the contents of the output window, the interpretation of each of the buttons, and the dynamic characteristics of the dialogue state. For example, upon invoking the function to present a message as scrolling text, the ‘start’ button may now operate as the ‘stop’ button, and after the message has finished scrolling, it may start again, or the dialogue may automatically change to a previous state or to some default starting point.

To design the presentation interface as a series of changes between window-definitions, a complete transition diagram of the dialogue is necessary. A rough sketch of the transition diagram results from using the entries in the dictionary of basic tasks as landmarks and from applying the function-to-button mapping principles. After this, the presentation design is a matter of filling in the details for each window and, whenever appropriate, adjusting the (e.g. automatic) transitions between dialogue states.

That functions are invoked by means of different dialogues may or may not be important to the specification of the production rules. For the purpose of designing the interface, the user-system dialogue should be completely specified, as described, from the dictionary of basic tasks, interaction principles, and some ‘syntactic sugar’. However, to specify the interaction in terms of the user’s competence knowledge, the production rule specification should not include elements of user control. The Comris wearable, as a highly interactive system in which the presentation of information by the system is interleaved with decision making by the user, shows the tension between ETAG as a flexible model for design representation and as a strict model of user competence knowledge. When ETAG is applied very strictly and user control is not allowed within basic tasks then additional tasks should be added, such as “change_mode” or “select_cancel”. For example, to specify that the user first has to select the proper mode, which may or may not already be the case, followed by selecting the intended task is simply treated as two different task invocations. In this case, the production rule specification of the wearable interface is similar to the specification of the kiosk interface. When ETAG is applied more loosely and some user control is allowed within basic tasks then composite production rules might be used. In this case, the example of selecting the proper mode and the intended task may be rewritten as a composite syntax-level production rule:

$$T[task] ::= \text{click\_button}[task] \ || \  
\text{click\_button}[select\_mode] + \text{click\_button}[task]$$
With multiple modes, multiple dialogue steps, or both at the same time, as is the case in three of the Comris wearable prototypes, the composite rules become fairly complex. However, using the strict ETAG interpretation, the different states of the system should be represented by preconditions in the event specifications, and the UVM should become more complex. A decision between the two options regarding a strict or relaxed interpretation can be made on empirical and on theoretical grounds.

Empirically, a decision between production-rule and UVM complexity depends on whether users perceive a difference between functional and interaction complexity. The usability experiment reported in De Haan (2000b) tried to answer one side of the question, hypothesising that a user interface with many dialogue steps would hide conceptual information whereas an interface which avoids dialogue steps by using control key combinations would provide a better overview of the functionality of the system. The results show that the two prototypes at the extremes of the dimension are about equally effective as well as about equally preferred. Given that the prototypes were not build to test the hypothesis but rather to pick the best one, we tentatively interpret the result as an indication of a trade-off relation between dialogue consistency on the one hand, and system overview and dialogue guidance information on the other.

From a theoretical point of view, a decision for the strict interpretation allows that the perceptual interface influences how users interpret the workings of the system. For design purposes, this should be avoided because the conceptual design of the user interface is a different phase from the perceptual design, which should be independent of each other, as far as possible (de Haan, 2000a). For modelling purposes, it should also be avoided because dependencies between the conceptual and the presentation levels of a model make it more difficult to interpret and apply. As such, even though the implication is that ETAG would no longer be used as a pure model of user competence knowledge, the more relaxed interpretation of the model is to be preferred for both design and for modelling purposes.

4. Discussion

In comparison to other design methods ETAG-based design requires designers to learn and use only one notation and, as such, it is relatively easy to use and provides more opportunities to allocate design resources to other issues, such as user interface adaptivity and adaptability (Oppermann, 1994). With respect to the facilities which ETAG-based design provides to represent user interface designs to accommodate diverse users, ETAG is able to represent different knowledge requirements about task performance between users and it is able to represent different ways of accessing a system in terms of characteristics of hardware devices and of interaction styles.

There are a more strict and a more relaxed interpretation of the ETAG notation with respect to representing the structure of the user-system dialogue in different systems or design alternatives. For both, design as well as modelling purposes, the more relaxed interpretation must be preferred, even though, introducing performance elements into the model is at odds with the original specification of ETAG. However, to decide between the strict and the relaxed interpretation on empirical ground, further research is necessary about the organisation of the user's knowledge about interactive systems and how different knowledge structures influence task performance.

References


