46

Automated Theory-based Procurement Evaluation

Andrew Howes, Stephen J. Payne, David Moffat

School of Psychology
Cardiff University of Wales, PO Box 901,
Cardiff CF1 3YG. WALES, U.K.
HowesA@cardiff.ac.uk

ABSTRACT In this paper we report progress on software for evaluating computer interfaces for the purposes of procurement. The software takes a description of the interface to be evaluated and outputs metrics describing the consistency of the interface. The output metrics are determined by automatically constructing a Task-Action Grammar from a manual description. We illustrate the evaluation tool with examples of evaluations of abbreviation structures and syntactic/semantic alignment.

KEYWORDS Metrics, Usability evaluation, Automatic tools, Task-Action Grammar, Consumer product

1. INTRODUCTION

Purchasers of new applications are often presented with a host of competitors, from which a choice must be made partly on the basis of which is easiest to learn and use. The most learnable and usable package will reduce training costs and, with a confident user community, be more effective during its life span. However, limited information is provided by manufacturers about the usability of their own software.

This information gap is partially bridged by trade magazines that review the usability and learnability of new software products (Bawa, 1994). Such magazines serve software purchasers who have become suspicious of the manufacturers' guarantee that their own software is easy to learn and use. We refer to this use of evaluation as procurement evaluation and contrast it to design evaluation, which has been much more widely studied in HCI. Ideally, procurement evaluation might be performed by purchasers, and therefore requires techniques that assume less expertise from the evaluator than does design evaluation. Procurement evaluation need only deliver summative information: It need not inform redesign, and therefore must report only on the quality of a system, not on the determinants of quality. Furthermore, despite its modest ambitions, successful procurement evaluation could have a long-term impact on the quality of design, because well-informed purchasers create an increased pressure on designers and manufacturers.

Consider an example of the procurement evaluation context: PC Magazine, which is published by Ziff Davis Ltd., and which advertises itself as a "lab-based magazine", prints reports on software usability compiled by their own in-house usability laboratory. In the March 1995 issue of this magazine there is a usability report on a number of available Personal Information Manager packages (PIMs) which provide diary-like functions. Each package is tested empirically by a number of evaluators and is scored on a number of indices of usability (page 178). Two of these indices are: productivity, or "the proportion of correctly-completed tasks measured against the time taken to finish them"; intuitiveness, which is "the effectiveness of the on-line help and the manuals, and how many references were made to them, measured against the number of calls to the supervisor."

We have been investigating how to automate the generation of indices similar to productivity and intuitiveness with a computer-based tool. Others have started to explore this idea (Barnard & May, 1993; Byrne, Wood, Sukaviriya, Foley & Kieras, 1994; Farenc, Liberati, Barthet, 1996). An automatic evaluation tool can be thought of as a function E that takes as input a description of the system, S, and outputs metrics indicating the usability and learnability of the system. In this paper we focus on measuring

Human–Computer Interaction: INTERACT'97 S. Howard, J. Hammond & G. Lindgaard (editors)
Published by Chapman & Hall

consistency (Payne & Green, 1986). Consistency is a measure of the degree to which the methods used to achieve goals share the same structure, and is one factor that influences the learnability of the device. Despite the amount of research that has been done on consistency new computer interfaces are still being produced with inconsistencies. For example, the Psion Organiser 3c used the keys "Psion-D" to change an entry in "Agenda" mode but in "Time" mode "Psion-D" means "delete". In this paper we specify E such that it gives a number C indicating how consistent the interface is, thus:

$$C = E(S)$$

where an improvement in the design of the interface would increase the value of C. (A more sophisticated version of the function would output further metrics.). In this paper we refer to three descriptions of the interface: The *input description* (S); the *intermediate description* (some psychologically motivated description of the knowledge required to use the device); and the *evaluation description* (C).

As Long (1989) and Barnard (1990) point out, work on interface evaluation in Human-Computer Interaction has tended to focus on specifying constraints on the intermediate description (e.g. Card, Moran, Newell, 1983; Payne & Green, 1986), rather than on determining how to deliver the psychological knowledge embedded within this description to the analyst.

The evaluation function must be both useful and usable for evaluators involved in the procurement process. To be usable, the input description needs to be readily available and the resulting evaluation description has to use simple and intuitive representational forms.

In this paper we report an evaluation tool that takes as input a description derived from a manual. Many manuals provide precise and formalised statements of how commands are achieved (e.g. the Unix command manual), or an on-line list of command specifications (e.g. Microsoft-Word provides a text-file listing of 337 commands, each with its menu location and keyboard accelerator).

To address usefulness we have based the evaluation function on Task-Action Grammars (Payne & Green, 1986, 1989). TAG is a formal notation for representing the knowledge necessary to use computer interfaces (Payne & Green, 1986, 1989 -see also D-TAG, Howes & Payne, 1990). The model uses semantic features within a rewrite grammar to expose how similarities and differences between interface tasks and methods can affect perceived consistency. TAG's sensitivity to the users' tasks means that it is not subject to the criticisms of Grudin (1989). By embedding psychological constraints within an intermediate description, TAG can achieve considerable savings in the expertise required for the inference of evaluation descriptions. However, the production of the rewrite grammar is itself a highly specialised and rather demanding skill. It is the process of producing a grammar that we describe how to automate in this paper.

The sections below illustrate two evaluation tools that we have built. The first introduces the general idea of automating evaluation and focuses on the evaluation of abbreviation consistency and performance time and the second looks at the evaluation of aspects of consistency that are dependent on the task semantics.

2. AUTOMATED EVALUATION

As an example of automated evaluation, consider the Unix C-shell. In this device the command to "remove a directory" is communicated by the user typing the character sequence "rmdir" and then typing the name of the directory. This method of communicating commands is not only used in command-oriented operating systems such as Unix and Ms-Dos, but also in text processing systems such as LaTeX, and to define "accelerators" in highly display-based (Graphic User Interface) environments like the Apple Macintosh and Windows. It is not, as is sometimes claimed, a mode of communication that has been entirely replaced by display-based devices. Moreover, the problem of designing consistent command syntax is not bound to keyboard-based inputs. It is also a potential problem for the design of voice activated systems and gestural interfaces, and is a very real problem for the design of consumer products such as Video Recorders. (There are many millions of Video Recorders, only a few are operated with a mouse.)

Here we talk through an evaluation scenario in which an evaluator compares the Unix commands for manipulating files and directories with the file manipulation commands for another (in this case

Interface: Unix (or NewUnix)		
Command Syntax	**Abbreviation**	**Words**
rmdir directory-name	rmdir	"remove directory"
mkdir directory-name	mkdir	"make directory"
cd directory-name	cd	"change directory"
ls directory-name (or lsdir)	ls (or lsdir)	"list directory"
more file-name	more	"more file"
rm file-name	rm	"remove file"

Table 1. The description of the (New)Unix interface(s) provided to the Evaluation Tool.

hypothetical) interface called NewUnix. The evaluation tool (written in Prolog) produces outputs that describe the consistency of the interface and the average performance time. The evaluation steps are:

(1) Provide a description of the relevant aspects of the interface. For our demonstration the description of the interface is provided in the form of the manual's description of the action sequence required for the command and the words that the command is an abbreviation of (see Table 1 for a description of some aspects of Unix).

(2) Run the evaluation function over the device description to learn that the average optimal performance time is 667 milliseconds and that the index of consistency is 1.2 (see Table 2 for an idea of how this information might be presented).

(3) Having evaluated the Unix interface, we can now move on to evaluating the NewUnix interface. This involves the same procedure. The description of NewUnix differs from Unix only in that the abbreviation for communicating the words "list directory" is "lsdir" rather than "ls" (see Table 1 again). This small change will illustrate the sensitivity of the consistency and performance time measures.

(4) The evaluation function can now be run over the NewUnix description. It says that the average optimal performance time is 767 milliseconds and that the index of consistency is 1.5 (Table 2).

(5) We have now generated two evaluations for two different interfaces: Unix and NewUnix. It can be seen from the output of the evaluation function that NewUnix (with a consistency coefficient of 1.5) is more learnable than Unix (with a consistency coefficient of 1.2) but that Unix (with an average performance time of 667 milliseconds) offers marginally more efficient performance times than NewUnix (with an average performance time of 767 milliseconds). This kind of trade-off is a common feature of interface designs.

These simple metrics raise many questions. For example, should the index of consistency be weighted for the frequency of use of the commands? We discuss these issues later.

Our prototype demonstrates the coarse properties of an automated evaluation tool, but no more. The important features to note are: (a) that a person using this tool need only specify the actions required for a command as they appear in the manual (grammatic analysis is not required), (b) that the tool in the form of a computer program performs an evaluation, and (c) that the output of the function can be quickly interpreted by someone who is not trained in psychology. Comparing consistency indices of 1.2 and 1.5 immediately reveals that the NewUnix interface is superior to Unix in this respect. We believe that indices of this form are

Interface Evaluator Report		
	Unix	NewUnix
Consistency coefficient	1.2	1.5
Average performance time	667ms	767ms

Table 2: Output of the evaluation tool for the two interfaces.

suitable for use in procurement evaluation and perhaps for publication in reviews such as those found in PC magazine.

3. SEMANTIC/SYNTACTIC ALIGNMENT

One of the most important features of the Task-Action Grammar (TAG) notation (Payne and Green, 1986, 1989) is that it can be used to evaluat semantic/syntactic alignment: The degree to which the different syntactic structures used in an interface correspond to the major semantic categories of the tasks.

The evaluation tool described in this section utilises this feature by constructing a database of TAG rules that capture how task-descriptions are mapped into the required action-sequences. A rule takes a set of semantic features of the task-description, and returns a sequence of syntactic tokens, i.e. a fully specified action-sequence. The consistency index is calculated on the basis of the number of rules required to represent all of the tasks.

One of the hardest problems that a human analyst faces when constructing a TAG is selecting semantic features to describe the tasks. The evaluation tool solves this problem by taking the English language sentence that is given as the manual definition of the command and using each word in the sentence as an individual semantic feature. We assume that the manual contains meaningful definitions and that the same words are used for the same concepts throughout the manual (the use of synonyms will cause the evaluation tool problems). In the current version, some simplification of sentences may be necessary before they are passed to the evaluation tool.

Rules are found from the set of manual-derived exemplars by examining their semantic and syntactic properties, and grouping the exemplars into classes that can be described by a single rule. The largest classes are preferred. In order to explain how this is achieved we introduce the concepts of a semantic characterisation and a syntactic class of commands. A set of semantic features is said to characterise a set of commands if those commands all share those features, and if no other commands share them. A general syntactic form can represent a set of commands if the syntactic form of each of those commands can be seen as a special case of that general form. For example, in Table 1, the commands that take a "directory-name" parameter are exactly the ones that are defined by the word / feature "directory". Mappings from a feature-set to a syntactic form are constructed only when a semantic characterisation and a syntactic class capture exactly the same set of commands, i.e. they are aligned.

The TAG constraints are not sufficient to specify a single grammar for a language: There are typically many grammars that satisfy its constraints. Here we assume that the most compact grammar for a set of tasks is the most useful for determining metrics.

In order to apply this assumption the evaluation tool needs heuristics to guide it to the most efficient grammar first. The prototype follows a locally-guided strategy which leads to an approximation of the most efficient grammar. The algorithm is:

1. Find the set CS1, that contains the largest number of examplars and that is semantically characterisable by some feature-set, call it FS1;
2. Check that there exists a syntactic form for every exemplar in CS1, call this syntactic form, SF1;
3. If (2) holds then:
   4. There exists a mapping rule FS1 → SF1;
   5. Remove the features that are common to all exemplars that are in CS1 and apply this algorithm to the remainder (recurse to step 1).
   6. Apply this algorithm to the commands that were not in CS1 (recurse to step 1).

7. else, if (2) does not hold then:
    8. Backtrack to step 1 and repeat the algorithm with the next-largest command-set, CS2.

In initial tests this algorithm has lead to a good approximation of the most efficient grammar.

The algorithm constructs rules that satisfy the TAG-constraints because of the checks in steps 1 and 2 that the set CS1 (or CS2 etc.) is simultaneously characterisable by the task semantics and by the command syntax. It also finds the most general rules because it tries the feature sets that characterise the largest number of exemplars first.

The algorithm terminates when there are no more commands in step 1 to be covered. The grammar is the collection of rules of the form FSi → SFi that were found each time the algorithm went through step 4.

To illustrate this algorithm we describe how it constructs a grammar for exemplars from an artificial interface called the "Toy Post Office" (Payne & Green, 1989). Payne and Green reported an experiment using three languages numbered 1, 2 and 3, in order of decreasing consistency. It turned out that Language 1 was easiest for subjects to learn, and is also describable (in a hand-written TAG grammar) in the fewest rules. While Language 3 was the hardest, and required several rules more at the highest level to describe it.

The "Toy Post Office" does not have a manual from which to select task descriptions. Instead we have used the English Language descriptions of tasks used by Payne and Green in their published paper. For example, Payne and Green (1989), page 219, state that to achieve the task, "high priority request for information about a missing brown boxer dog" then the user should type "DOG BOXER HIGH BROWN."

Each of the three grammars took the evaluation tool about 10 seconds to generate on a 486-PC running SWI-Prolog on the Linux OS. As expected, the grammars were similar to those written by Payne and Green. They only differ where the evaluation tool is unable to handle null-rewrites (where a sub-phrase may sometimes be allowed to be empty).

In describing how the tool constructed the grammars we look only at the second half of the languages (commands s17 to s32) and we only compare languages 1 and 2. We will not explain what the tasks achieve, we are only interested here in the relationship between their semantics and their syntax.

In Table 3 the task-description given to the evaluation tool for command s28 was: "Change person X's password to Y, with low priority." The program takes all of these words to be semantic features. Notice that the task features are not identical to their action tokens. The program determines correspondences by looking for configural alignments in the whole set of exemplars. For example, the feature "person" goes with the token name, because they both occur in the same command set. Also notice that apparently superfluous features, which occur in every command, like "priority," are ignored because they do not serve to distinguish between commands. These features, therefore, only appear in a rule if the entire set of commands is syntactically unifiable, in which case one can say that they do indeed turn out to characterise a degenerate command-set: namely, the whole language.

The right-hand part of Table 3 shows the grammar deduced by the consistency evaluator. The top of the grammar is 'sentence', which expands via one rule into a sequence of subtasks, such as 'sub1(PsF1)'. The subtasks themselves expand into tokens. Alternative expansions of the subtasks are possible. These are numbered. The variable 'PsF1' occurs twice in the 'sentence' rule, which means that the expansion chosen for 'sub1' should agree with the expansion for 'sub3'. Thus, the grammar does not allow commands in which 'job(X)' occurs in the first position and 'password(Y)' in the third, because that would require the 'PsF1' variable to be simultaneously bound to the constants '4' and '3' in the same expansion. A particular variable can only be bound to one constant in a particular instantiation.

The variable 'PsF2' occurs uniquely in the top-level rule (for 'sentence'), so that there are no binding restrictions on it. Therefore any priority can occur embedded between any possible first and third tokens.

The variables (e.g. 'PsF1') are what we call pseudo-features. They are synthesised by the program as it builds the grammar, simply to support the binding restrictions we want in order to enforce co-occurrences of the right terms with each other. How this synthesis is accomplished is not so important here, but we do view these pseudo-features as a minor inconvenience. Our aim is that the evaluation tool should use the task-description derived features, that constrained the grammar construction, in the output.

Having constructed a grammar the evaluation tool calculates a complexity metric for the grammar. The

Sentences of Language 1

s17. bluff, very_high, bluff_game_type(X)
s18. bluff, very_low, bluff_game_type(X)
s19. bluff, high, bluff_game_type(X)
s20. bluff, low, bluff_game_type(X)
s21. quote, very_high, quote_game_type(X)
s22. quote, very_low, quote_game_type(X)
s23. quote, high, quote_game_type(X)
s24. quote, low, quote_game_type(X)
s25. name(X), very_high, password(Y)
s26. name(X), very_low, password(Y)
s27. name(X), high, password(Y)
s28. name(X), low, password(Y)
s29. job(X), very_high, town_office(Y)
s30. job(X), very_low, town_office(Y)
s31. job(X), high, town_office(Y)
s32. job(X), low, town_office(Y)

Grammar

sentence → sub1(PsF1) sub2(PsF2) sub3(PsF1)

sub1(1) → bluff
sub1(2) → quote
sub1(3) → name(X)
sub1(4) → job(X)

sub2(1) → very_high
sub2(2) → very_low
sub2(3) → high
sub2(4) → low

sub3(1) → bluff_game_type(X)
sub3(2) → quote_game_type(X)
sub3(3) → password(Y)
sub3(4) → town_office(Y)

Table 3. Language 1 and its deduced grammar.

current metric is very simple: Count the alternatives in the top-level rule. In the case of the grammar in Table 3, there is just 1 alternative.

Language 2 is made more complex than Language 1 by the different position of the priority tokens in some of the command (see Table 4). The evaluation tool fails to find a syntactic form that covers all of the sentences. Instead, it breaks them into two groups on the basis of distinguishing task features, and finds a rule that fits each group. As a consequence, the grammar in Table 4 has two alternatives in its top-level rule instead of Language 1's single possible expansion for 'sentence'. According to our simple metric, therefore, Language 1 is the better than Language 2. The predictions made by the evaluation tool thereby match Payne and Green's prediction, which is supported by their data.

The figures arrived at by the evaluation tool summarise the consistency scores of the three whole languages as follows. We take consistency to be inversely proportional to the size of the languages optimally small rule-set, as suggested by Payne & Green:

Language 1: C = 1/2 = 0.50
Language 2: C = 1/3 = 0.33
Language 3: C = 1/7 = 0.14

These indices or metrics could be used in a procurement evaluation context to determine which interface is best on this dimension. We emphasise that they have been produced automatically using a program that works from manual-like descriptions and that they are sensitive to configural aspects of the semantics of the tasks being analysed.

4. DISCUSSION

We have described in general terms what automated theory-based procurement evaluation might mean and in addition we have described Prolog implementations of consistency evaluators that take manual-like descriptions as input. We have demonstrated that an automatic evaluation tool can derive representations, closely related to Task-Action Grammars, that previously had to be written by hand, and in addition

Sentences of Language 2

s17. very_high, bluff, bluff_game_type(X)
s18. very_low, bluff, bluff_game_type(X)
s19. high, bluff, bluff_game_type(X)
s20. low, bluff, bluff_game_type(X)
s21. very_high, quote, quote_game_type(X)
s22. very_low, quote, quote_game_type(X)
s23. high, quote, quote_game_type(X)
s24. low, quote, quote_game_type(X)
s25. name(X), very_high, password(Y)
s26. name(X), very_low, password(Y)
s27. name(X), high, password(Y)
s28. name(X), low, password(Y)
s29. job(X), very_high, town_office(Y)
s30. job(X), very_low, town_office(Y)
s31. job(X), high, town_office(Y)
s32. job(X), low, town_office(Y)

Grammar

sentence → sub1(PsF1) sub2(PsF2) sub3(PsF2)
sentence → sub4(PsF1) sub1(PsF2) sub6(PsF1)

sub1(1) → very_high
sub1(2) → very_low
sub1(3) → high
sub1(4) → low

sub2(1) → bluff
sub2(2) → quote

sub3(1) → bluff_game_type(X)
sub3(2) → quote_game_type(X)

sub4(1) → name(X)
sub4(2) → job(X)

sub6(1) → password(Y)
sub6(2) → town_office(Y)

Table 4. Language 2 and its deduced grammar

we have shown that the evaluator predicts the learnability of previously reported experimental results.

A potential problem with the evaluation tool arises from the fact that it is essentially a hill-climber, and as a consequence it can be trapped by local maxima. Whilst the algorithm has produced good results on small toy problems we are currently investigating whether it scales up to full application-size sets of commands (> 200).

In addition, there are a number of issues with the evaluation description (the metrics given as output) that interest us. The productivity and intuitiveness indices used by PC magazine have given us an initial guide as to what is required. However, there are many issues unanswered. For example, (a) Should task-action mappings that are used more frequently have a larger effect on the overall index? It might be that it is more important that infrequently used commands are consistent so that they can be readily inferred. (b) Should the indices be affected by the size of the interface? For example, should a device that has 300 commands and only two syntactic forms be rated as more consistent than a device that has 10 commands and two syntactic forms? (c) Is it useful for the tool to provide different metrics at different levels of analysis? The prototype described above gives only the top-level of summary analysis. At the next level, consistency might be unpacked to reveal particularly problematic commands (hot spots; Howes & Young, 1991, 1996), and performance time might be unpacked to give worst-case and best-case tasks. (d) Will it be useful to determine some of the indices relative to larger scale benchmark tasks (e.g. write a letter) rather than the simple-tasks (e.g. open a file) considered in the prototype tool?

5. REFERENCES

Barnard, P.J. (1990) Bridging between basic theories and the artifacts of human-computer interaction. In J.Carroll (Ed) *Designing Interaction - Psychology at*

the Human-Computer Interface. Cambridge: Cambridge University Press.

Barnard, P.J. & May, J. (1993) Cognitive modelling for user requirements. In P.F. Byerley, P.J. Barnard & J.May (Eds) *Computers, Communication and Usability: Design Issues, Research and Methods for Integrated Services*. Elsevier Science Publishers B.V.

Bawa, J. (1994) Comparative usability measurement: the role of usability lab in PC Magazine UK and PC/Computing. *Behaviour and Information Technology, 13*, 1 & 2, 17-19.

Byrne, M.D., Wood, S.D., Sukaviriya, P., Foley, J.D., & Kieras, D.E. (1994) Automatic interface evaluation. In *Proceedings of Human Factors and Computing Systems: CHI'94*. Boston, April, 1994.

Card, S.K., Moran, T.P., & Newell, A. (1983) *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Farenc, C., Liberati, V., & Barthet, M. (1996) *Proceedings of CADUI'96 2nd International Workshop on Computer-Aided Design for User Interfaces* - Namur, Belgique, June 1996.

Grudin, J. (1989) The case against user interface consistency. *Communications of the ACM, 32*, 1164-1173.

Howes, A. & Payne, S.J. (1990) Display-based competence: Towards user models for menu-driven interfaces *International Journal of Man Machine Studies, 33*, No. 6, 637-655.

Howes, A. & Young, R.M. (1991) Predicting the learnability of task-action mappings. In S.P. Robertson, G.M. Olson, J.S. Olson (Eds.) *Reaching Through Technology - CHI'91 Conference Proceedings: Human Factors in Computing Systems*, New Orleans, Louisiana. Addison-Wesley. 113-118.

Howes, A. & Young, R.M. (1996) Learning consistent, interactive and meaningful device methods: a computational model. *Cognitive Science, 20*, 301-356.

Long, J. (1989) Cognitive ergonomics and human-computer interaction. In J.Long and A.Whitefield (Eds.) *Cognitive Ergonomics and Human-Computer Interaction*. Cambridge University Press.

Payne, S.J. & Green, T.R.G. (1986) Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction, 2*, 93-133.

Payne, S.J. & Green, T.R.G. (1989) The structure of command languages: an experiment on task-action grammar. *International Journal of Man-machine Studies, 30*, 213-234.