DOCUMENTATION SKILLS IN NOVICE AND EXPERT PROGRAMMERS: AN EMPIRICAL COMPARISON

Jean-François Rouet. Catherine Deleuze-Dordron and André Bisseret
National Institut for Research on Informatics and Automation (INRIA)

INRIA
Language and Communication Lab.
95 avenue du Recteur Pineau
F-86022 Poitiers Cedex
rouet@isis.imag.fr
email rouet@isis.imag.fr

Paper proposed to
Psychology of Programming Interest Group Meeting (PPIG)
Edinburg, Scotland, 4-6 January 1994

extended abstract

1. Introduction

This study is part of a broader project which aims at studying the use of natural language documentation in design activities. A particular objective is to identify the cognitive factors that may influence description of software components.

Software design problems often involve ill-specified objectives and constraints. Moreover, several solutions may be proposed to solve a given problem. Thus, software design requires a planning activity that leads to the progressive specification of the relevant problem space (Newell and Simon, 1972), until implementable solutions can be defined. Previous research has evidenced that the planning involved in software design is opportunistic (Hayes-Roth and Hayes-Roth, 1979), i.e. the designer may depart from the original decomposition scheme, as a function of intermediate results, unexpected difficulties, and information found *en route* (e.g., Guindon, 1990; Hoc, 1988).

Documentation plays an important part in design activities, however its relationships with design problem solving are still obscure. A series of preliminary studies (Deleuze-Dordron, 1993) suggested that comments inserted into programs may vary with respect to their role (e.g., describe vs. explain the program) and the level of entities commented (high vs. low level entities). Moreover, expert designers seem to possess specific criteria when evaluating the relevance of comments (e.g., is the comment redundant?).

In the present study we wanted to characterize the influence of design expertise on the type, location and strategy of production of inserted programs. A general hypothesis is that comments reflect the designer's cognitive representation of the entity being commented. The cognitive representation may vary depending on the context in which an entity is being examined, whether or not the context provides meaningful information about the role or purpose of the statement.

The program representation is also influenced by the designer's expertise: Novices tend to be influenced by surface features of the program, whereas expert designers organize their representation according to the underlying solution scheme. Consequently, we hypothesized that expert designers may be more able to exploit a meaningful context when commenting a piece of software.

Rationale of the experiment. We designed an experiment in order to check the following interaction hypothesis: Novices and experts would produce merely descriptive statements when commenting isolated statements; When presented with statements within a meaningful context (procedures or whole program) expert designers would produce more semantic explanations (i.e., comments about the solution being implemented) than novices.

2. Method

Materials. We used a set of programs and program excerpts in ADA based on several programming manuals. The excerpts were chosen so as to be understandable by novice and expert programmers. The final sample was made of a series of 20 isolated statements, four isolated procedures and two short programs (53 and 62 statements).

Subjects were 27 programmers at two levels of expertise: Novices (Freshmen from a two-year computer science program, N=20) and experts (faculty members and industrial designers, N=7). The terms "novices" and "experts" are used for purpose of clarity although the subjects' actual level of skill was heterogeneous both in the novice and the expert group.

Procedure: The experimental task consisted in commenting ADA statements "*so as to make the program more understandable by programmers at your level*", in three conditions or tasks:

Task 1 - Isolated statements: Subjects were asked to select and comment 10 statements among 20 presented.

Task 2 - Isolated procedures: Subjects were asked to insert 5 or 6 comments in each of four procedures (two of which were common with task 3).

Task 3 - Simple programs: Subjects were asked to comment two simple programs.

Each subject was asked to perform the three tasks. Novices participated in one collective session. Experts were run in small groups in the course of several sessions. Completion time was identical for all subjects.

Dependent measures concerned the choice of statements commented and the nature of the issued comments. The experimental hypothesis was that experts would produce more explanations on high level entities than novices, but only when a meaningful context was available.

3. Results

At the present stage data are still being analyzed and only partial results will be reported in the present abstract. For this reason, no statistical generalization can be made.

Content analysis of the comments: Based on an informal examination of the protocols we built up a categorization framework including five main categories (Table 1).

Table1: Analysis framework used to score the comments.

Category	Definition	Example	
		Statement	Sample comment
1- PARAPHRASE	Comment paraphrases program statement and does not include any new information.	i:=i+1	*"i is incremented by 1"*
2- LABEL SEMANTICS	Comment based on meaning an entity label	if prime (nber, beg) then	*"if number is prime then"*
EXPLANATION	Comment includes information...		
3- SYNTACTIC	- about programming rules.	current:= current.next;	*"going through the list 'courant'"*
4- SEMANTIC	- about solution being implemented.	cont_dig:= codeid mod 10:	*"extraction of last digit of code"*
5- META-COMMENT	Statement about commenting.	function PGCD (k, 1:integer).	*"describe functioning and parameters"*

Definitions presented in Table 1 were made operational with a number of scoring rules. Then the scoring framework was refined iteratively. Finally the framework was tested for reliability on the basis of a sample of protocols. An independent double scoring showed an inter-rater agreement higher than 80%.

Effects of context and expertise. Figure 1 shows the main categories of comments issued by novices and experts.

Insert Figure 1 about here

When no context was available, novice programmers issued mostly paraphrases of the presented ADA statements (see Table 1). In contrast expert programmers also issued other types of information, including syntactic explanations and meta-comments.

When a restricted context was available, the frequency of semantic explanations increased in large proportions. The increase was larger in the expert group.

Finally, when a full context was available both novices and experts issued mostly semantic explanations.

The results did not exactly confirm our initial hypothesis. We observed indeed that there was an increase in semantic explanations when a context was made available. However the main difference between novices and experts occurred when no context was available. Apparently experts were able to draw syntactic explanations from their general knowledge of the language. They also made comments about what they might say if a context was made available (see Table 1, "meta-comments"). Novices merely re-phrased the statement, with very little elaboration or contextual information.

Experts were also better at taking advantage of a limited context. When presented with a single procedure, they were able to issue semantic explanations of the solution scheme underlying the procedure. Novices were also able to do so, although to a lesser extent.

Finally, when asked to comment a full program, both novices and experts managed to provide semantic explanations. It should be noticed that in the present case the programs were quite simple and could be understood even by beginners. In fact, we obtained preliminary evidence that when the problem was less familiar, the proportion of semantic explanations tended to decrease: Task 2 (partial context) included a procedure implementing a concrete problem (isolate figures in a Roman number) and a procedure implementing a string problem (check if a number is "perfect", a simple but unfamiliar notion as evidenced by a pre-test). The types of comments issued for these two procedures are shown in Figure 2

Insert Figure 2 about here

As indicated in Figure 2, the percentage of semantic explanations was higher for the familiar procedure in both groups. From these partial results it may be suggested that the effect of domain expertise is distinct from the effect of general software design expertise. In other terms even experienced programmers may have trouble commenting a program if they do not have a good understanding of what it does.

Finally we looked at the entities commented when a full context was available (task 3). Results are presented in Figure 3.

Insert Figure 3 about here

As shown in Figure 3, comments were not distributed randomly throughout the program. "Structural" statements e.g., procedure declarations, beginning of loops and function calls were most frequently commented. input and output received less comments. Finally, "end" and "begin" instructions (which are redundant in ADA) were seldom commented. Figure 3 also indicates that the experts tended to produce more comments than the novices, especially for statements concerning local structures (LOOPs, CALLs, EXITs). However at the present stage there is no clear evidence for an interaction.

4. Discussion and conclusion

The main purpose of this experiment was to investigate the role of expertise in the production of computer program comments

The analysis of comments elicited several categories of information: paraphrases, syntactic and semantic explanations, meta-comments and inferences from labels. Overall paraphrases were more frequent in novices which support previous findings that novices are influenced by surface features of the program. The experts issued more explanations and other comments regardless of the task.

Although we did not observe the expected interaction, we did find a relation between expertise and context of production. The main novice-expert difference occurred when no context or only a limited context was available. When the full program was available, both novices and experts issued merely semantic explanations, and their did so mostly for program statements high in the hierarchy. The only difference then was that experts tended to produce a larger number of comments.

The observed novice-expert differences are of special interest given the large heterogeneity within each group: Some novices had quite a lot of experience in at least one programming language; the experts came from very different areas of activity. Despite these differences, we were able to observe different patterns of commenting across groups.

This experiment supports the hypothesis that there is a tight relationship between the cognitive representation of a program and the type of comments provided. This finding suggests that producing quality documentation is not just a matter of being careful or rigorous, but may require specific forms of expertise.

We are currently in the process of analyzing in more detail the entity-comment relationship. Subjects' opinions and self-reports about documentation usage will also be considered.

The experimental data also have to be checked against actual documentation strategies in the course of real life design projects. In this perspective we have conducted a case study of documentation during design with reuse, in cooperation with a team of expert designers (Rouet et Deleuze-Dordron, 1994). It might also be of interest to related documentation in software design with information usage in other areas of expertise (e.g., architectural design).

Natural language documentation is a central component of design activities which has been rather overlooked so far. Further research on documentation skills may contribute to a comprehensive model of design.

Acknowledgements

This study was supported as part of the ESPRIT-SCALE project. The authors wish to thank the students and staff at IUT de Valence (France) as well as the industrial designers for their contribution to this study.

Références

Deleuze-Dordron, C. (1993). *Analyse de l'activité de documentation de programmes: première exploration.* Mémoire pour le DEA de Sciences Cognitives, Grenoble: INPG, 1993.

Guindon, R. (1990). Designing the Design process : Exploiting Opportunistic Thoughts. *Human Computer Interaction, 5,* 305-344.

Hoc, J.M. (1988) - *Cognitive psychology of planning.* London: Academic Press.

Rouet, J.-F. & Deleuze-Dordron, C. (1994). *Design with reuse and software documentation: Some cognitive and human factors issues.* INRIA Rhône-Alpes, working paper.

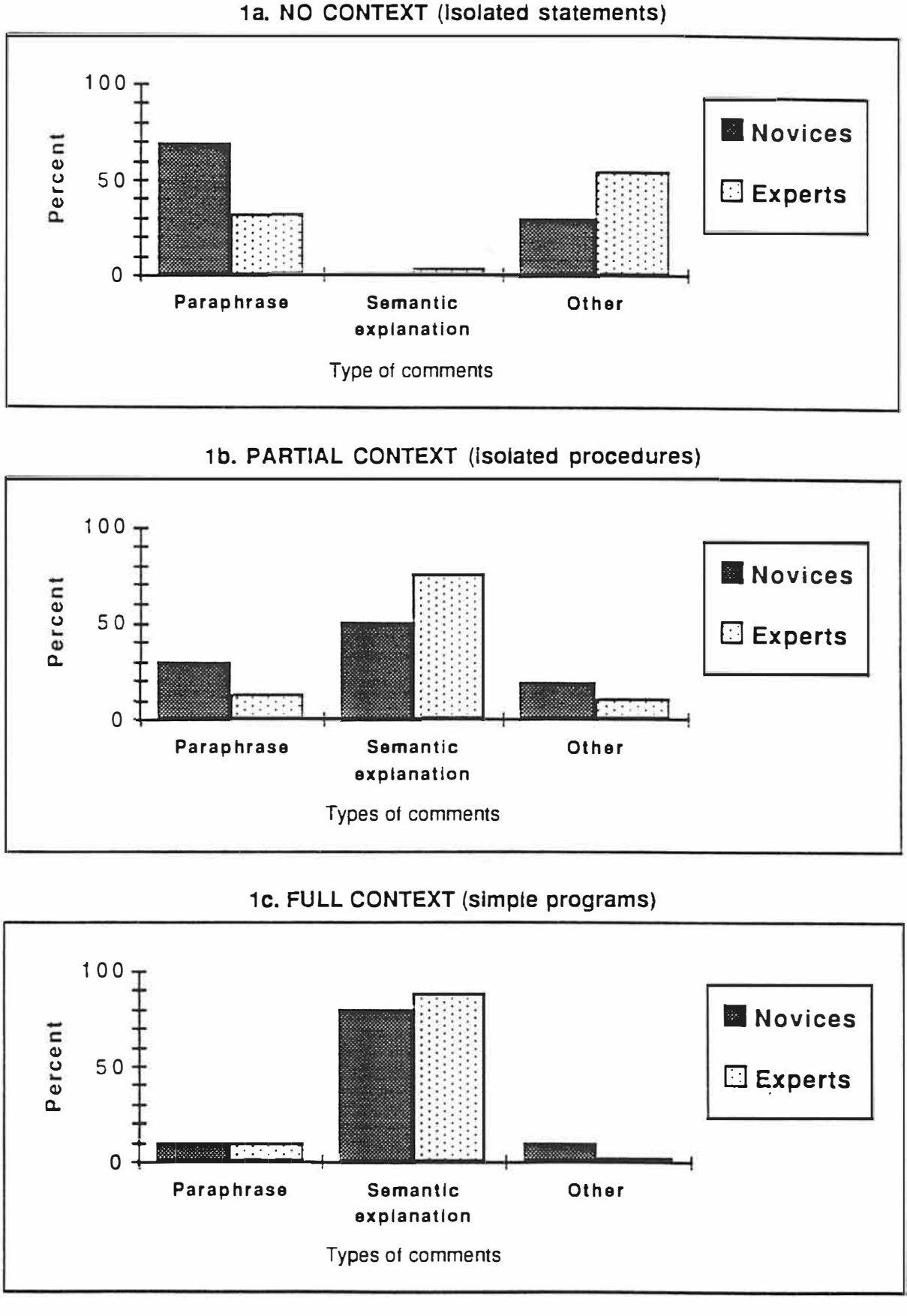
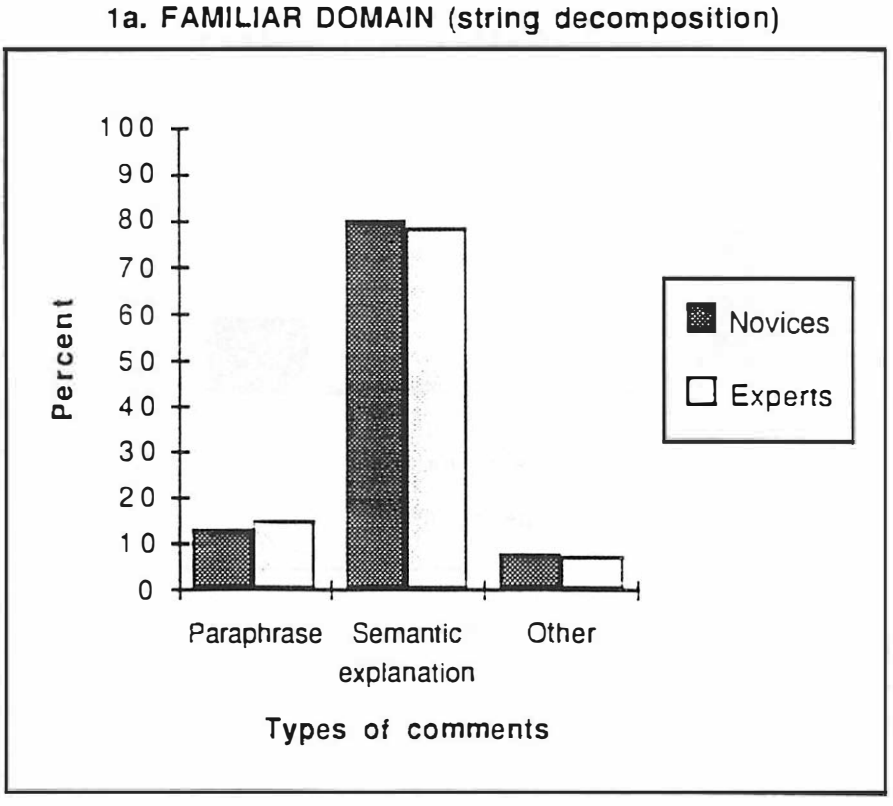


Figure 1: Types of comments as a function of context and expertise level.
NB. For partial and total context, only common procedures have been included.



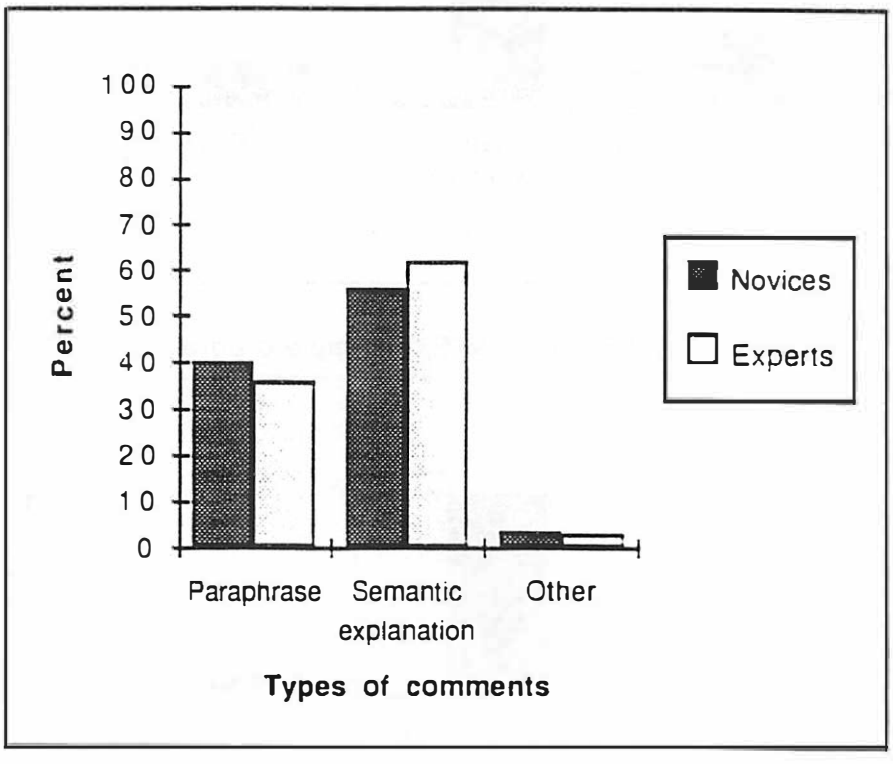
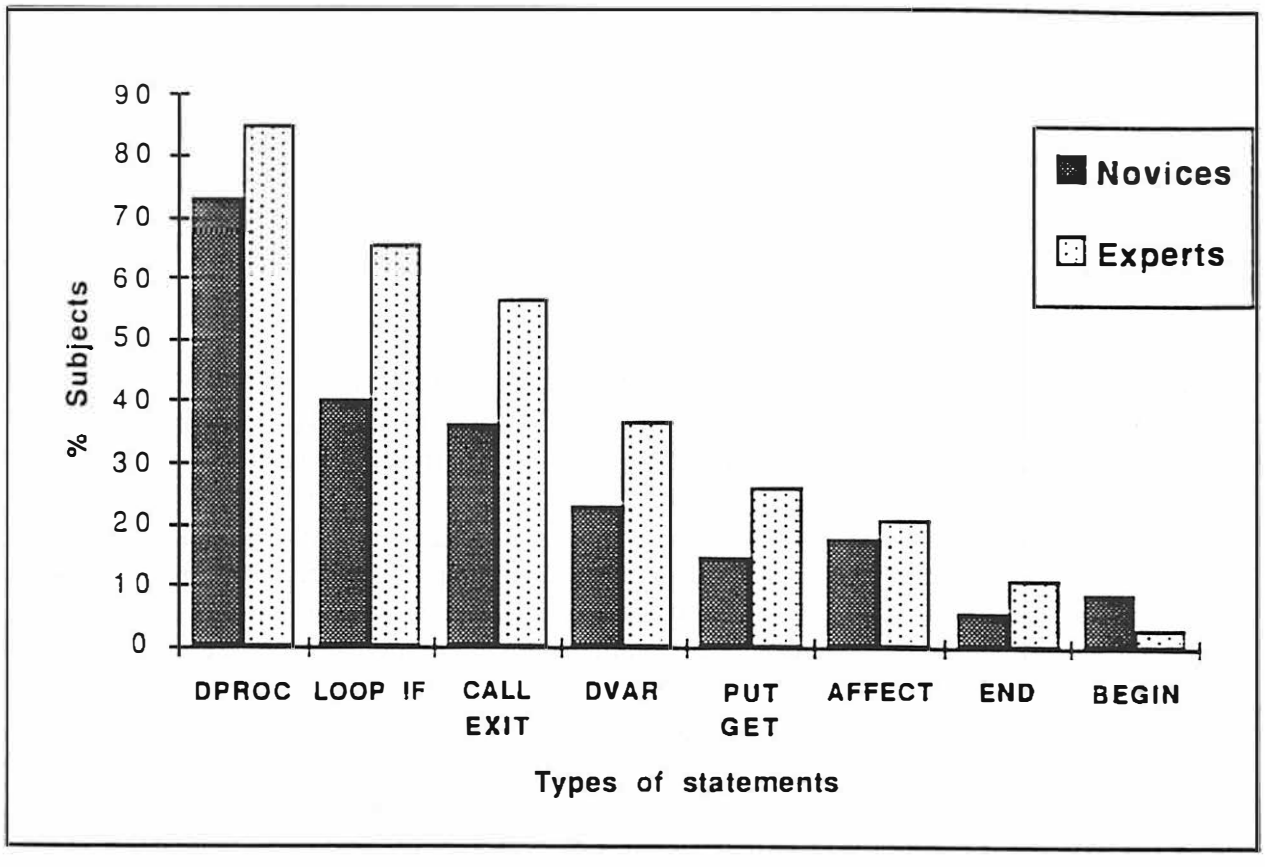


Figure 2: Types of comments as a function of domain familiarity (task 2, isolated procedures #3 and 4).



DPROC: Declaration of procedure, function ou package
LOOP-IF: Beginning of local block
CALL-EXIT: Call or exit of procedure
PUT-GET: Input-output
AFFECT: Allocation of value to variable
DVAR: Declaration of variable or type
END: End of block
BEGIN: beginning of bloc

Figure 3: Frequency of comments (% of subjects) as a function of type of statement (total context, mean frequency for programs 1 et 2).