Compiler Error Messages: What Can Help Novices?

Marie-Hélène Nienaltowski
School of Computer Science and Information Systems
Birkbeck, University of London
Malet Street, Bloomsbury
London WC1E 7HX
marie-helene@dcs.bbk.ac.uk

Michela Pedroni
Chair of Software Engineering
ETH
8092 Zurich, Switzerland
michela.pedroni@inf.ethz.ch

Bertrand Meyer
Chair of Software Engineering
ETH
8092 Zurich, Switzerland
bertrand.meyer@inf.ethz.ch

ABSTRACT
Novices find it difficult to understand and use compiler error messages. It is useful to refine this observation and study the effect of different message styles on how well and quickly students identify errors in programs. For example, does an increased level of detail simplify the understanding of errors and their correction? We analyzed messages produced by a number of compilers for five programming languages, and grouped them into three style categories from their level of detail and presentation format, and correlated the level of experience and error type with performance and speed of response. The study involved two groups of students taking an introductory programming course at two different institutions; they used messages in these three styles to debug erroneous code. The results indicate that more detailed messages do not necessarily simplify the understanding of errors but that it matters more where information is placed and how it is structured.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education;*

General Terms
Experimentation, Human Factors.

Keywords
Compiler error messages, novice programmers.

1. INTRODUCTION
As many teachers of programming have noted [6], commercial compilers are built for experts. In tutoring systems for programming, there are currently three main approaches to improve compilers for novices: writing new compilers specifically devised for such users [7]; improving existing commercial compilers [4]; enhancing the error messages (for example by rewriting them in layman's terms [1]). What matters to students is not the compiler used but the messages. Why are those from commercial compilers not good enough for novices? Reasons are numerous: they are too brief, not visual enough, too technical, etc.

SIGCSE'08, March 12–15, 2008, Portland, Oregon, USA.

In this work we analyze these observations more systematically by studying experimentally how message style affects novices. The assumption is that longer explanations and suggestions of error corrections improve novices' understanding of the problem and therefore their performance. The study used a multiple-choice questionnaire that combined three error message styles with three error types in nine questions. Each of the nine questions required the student to identify an error in a program extract, from the message and the extract itself. The three styles are a synthesis of the techniques used in both commercial compilers and compilers purposefully built for novices. The choice of the three error types relies on earlier publications on common novice errors [2] and data collected from our own students over the past three years [5]. It includes common novice errors: unknown identifier, wrong number of arguments, private access violation. Two groups of students studying two different programming languages at two different institutions answered the questionnaire. One group answered online, allowing tracking of answering time; the other answered the questionnaire in class in limited time. We present the setup, analyze the collected data, and discuss the results.

2. COMPILER ERROR MESSAGES
A survey of existing error message styles led to the derivation of three broad styles. Table 1 shows the compilers used in the survey. The styles are: a short form of error message, a visually inline form, and a long form.

Table 1: Surveyed compilers

		Short form	***Visual form***	***Long form***
Programming Environment	***C++***	Digital Mars, Borland, GNU C++, IBM XL C/C++ for AIX		
	Java	SUN JDK6	BlueJ	
	Ada	Gnat-ada95		
	Scheme		Dr. Scheme	
	Eiffel			EiffelStudio

The **short form** is most commonly used in commercial compilers. It is displayed separately from the code (e.g. in a console or a separate part of a programming IDE) and consists of the file name and line number where the error occurred, type of the error, a brief error message, and a code snippet.

Short form example:

```
ticket_machine.e, line 27: Cannot find identifier.
total := total + price
^
```

The **visually inline form** highlights the line where the error occurred in the code and gives a brief error message.

Visually inline form example:

```
class TICKET_MACHINE
feature {NONE} -- Access
   price: INTEGER
       -- Cost of ticket
   balance: INTEGER
       -- Amount of inserted money
   total: INTEGER
       -- Total value of transaction

feature -- Basic Operations
   print_ticket is
       -- Print ticket.
     local
       an_amount: INTEGER
     do
       if ballance >= price then
         io.put_string ("USD " + price.out)
         total := total + price
         balance := balance - price
       else
         ...
       end
     end
end

Error message:
Cannot find identifier.
```

The **long form** consists of an error code, brief error description, suggestion of what to do, affected class, affected feature, involved token, line number, and a code snippet of where the error occurs.

Long form example:

```
Error code: VEEN
Error: unknown identifier.
What to do: make sure that identifier, if needed,
is final name of feature of class, or local entity
or formal argument of routine.
Class: TICKET_MACHINE
Feature: make_tm
Identifier: price
Taking no argument
Line: 10
do
-> price := ticket_cost
balance := 0
```

Some compilers display a list of error messages if more than one error exists in the code. To facilitate the study and for consistency, the questionnaire presented one error at a time; it has been argued anyway that this is better for novices [1].

3. STUDY SETUP

Two groups answered questions that combined the three error message types applied to three common types of novice errors.

3.1 Questionnaire

The questionnaire consisted of nine questions representing the combination of three message types and three error types as summarized in Table 2. For example, $Q4_{s\_id}$ represents Question 4 relating to the *Unknown identifier* error and using the short form message. Each question presented one or two classes (as the example in Section 2.1.2 shows) with an error in the code. One of the message types was shown, followed by a multiple-choice question asking the student to identify the error. Published studies on the most common error messages [2] and data collected from our students over the past three years [5] guided the choice of error types, which included:

Unknown identifier. In all studies this error type is the most common novice error.

Wrong number of arguments. This is the second most common error type in our data collection.

Private access violation. This error appears in our own data collection and in [3] as one of the top 12 errors.

The questionnaire used error types of differing complexities to ensure that variations (or their absence) in the results would not be due solely to the choice of the error types, thus allowing generalizations of the findings for all types of errors. The *Unknown identifier* error was the least complex and involved only one class in the questions. The *Private access violation* error was the most complex since it required students to identify how the two presented classes interacted.

Table 2: Questions: combinations of message and error types

	Short form (s)	**Visual form (v)**	**Long form (l)**
Unknown identifier (id)	$Q4_{s\_id}$	$Q5_{v\_id}$	$Q2_{l\_id}$
Wrong number of arguments (arg)	$Q9_{s\_arg}$	$Q6_{v\_arg}$	$Q8_{l\_arg}$
Private access violation (acc)	$Q1_{s\_acc}$	$Q3_{v\_acc}$	$Q7_{l\_acc}$

3.2 Participants

From the 2006-2007 Introduction to Programming course at ETH, 43 students answered the questionnaire. Students of this course typically come with various backgrounds; few are completely new to programming. The course uses Eiffel. In addition to fundamental OOP and procedural concepts such as objects, classes, inheritance, control structures, recursion, it covers more advanced topics such as event-driven and concurrent programming and fundamental concepts of software engineering. The questionnaire was accessible online[1] towards the end of the semester to all students wishing to participate in the study. In background questions, the participants rated their level of programming expertise on a Likert scale of 1 to 5 (where 1 represented having little experience and 5 a lot). 30% described themselves as belonging to levels 1 and 2; 37%, 28% and 5% as at levels 3, 4 and 5 respectively, as illustrated in Figure 1. The students answered the compiler error questions one at a time in a preordered sequence. The time at which the student started a question was recorded. Students had 45 minutes to complete the questionnaire. The timing data indicates students spent the time answering questions on task. Most completed the questionnaire before time-out.

From Birkbeck, University of London, 24 students studying the Software and Programming 1 module answered the questionnaire. The course uses Java and covers the basics of programming such as loops, selection, assignments and basic concepts of object oriented programming such as classes and objects, feature calls, argument passing; it does not go as far as inheritance. In contrast to ETH, most of the participants in that course have never

[1] at http://se.ethz.ch/people/pedroni/compilererrors

programmed before. 75% describe themselves as being at level 1, the rest do not go beyond level 4 (Figure 1). Volunteers had one hour to complete a paper questionnaire in class towards the end of the term in 2007. It used the same questions as the ETH questionnaire, but adapted to Java terminology. It was not possible to control the order of question answering although the instructors asked students not to go back to previous questions once they had completed them. This setup also prevented recording of times as for the ETH students.

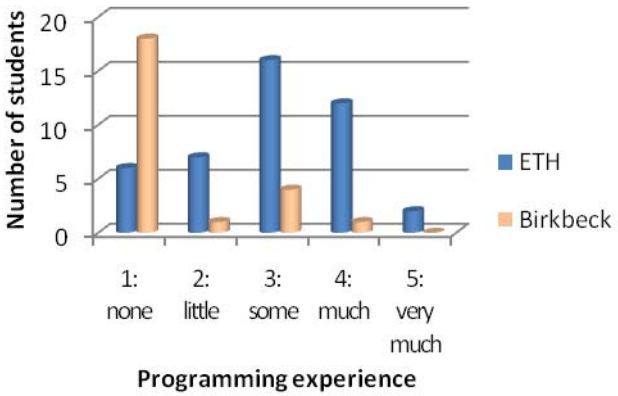

Figure 1: Experience levels of the two groups

4. ANALYSIS

The principal conjectures under evaluation are: (1) the more information a message provides, the more likely the student (irrespective of his level of experience) is to understand the message, identify the error and suggest an appropriate correction; and (2) novices, in particular, benefit from enriched messages since they get their answers right as they obtain more information on the error and explanations of how they could correct it.

Additional conjectures included whether an enriched message results in shorter response times independent of the correctness of the student's answer and of their level, and whether the error type determines the number of correct answers.

The analysis treats the results separately for the two different groups. For analyses where time is involved, only the online questionnaire filled in by the ETH students provides information.

The first limitation of this study is the possibility of self-selection introduced by voluntary participation. Additionally, since questions were lengthy, the questionnaire was designed with only one question per pair of error message style and error type. Having more questions per pair could help leveling out differences in the level of difficulty for single questions. Adding error types might also have helped in producing clearer results. These two limitations are threats to the validity of these results.

5. RESULTS

This section details the results obtained for each of the six stated hypothesis. It describes the motivation for the hypothesis, the results for the ETH group and their interpretation, then the results and interpretation of these for the Birkbeck group, and the results overall.

A general observation is that students from ETH generally did much better than the Birkbeck students, reflecting their more advanced level. This is also reflected in their self-assessment of their level of knowledge (see Section 3.2).

Hypotheses A to C measure correlations between programming experience and the outcomes of the questionnaire. They answer the following questions: Does the programming experience significantly influence the time needed to answer questions? Does the programming experience significantly influence the number of correct answers? And does the programming experience still significantly influence the number of correct answers if discriminating according to message styles? The analyses use the Spearman's rank correlation to assess the direction and strength of the relationship between the two variables under test.

Hypotheses D to F compare means for the outcomes of the study discriminated by a second variable. They answer the following questions: Does a certain message style produce significantly more correct answers? Does a certain message style result in significantly faster answers? And to complement the findings, does a certain error type produce significantly more correct answers? These analyses use the Wilcoxon rank-sum test, a non-parametric alternative to the two-sample t-test, and assess whether two samples of observations come from the same distribution.

Hypothesis A: Higher experience results in faster answers.

Someone who has significant experience and encounters an error will have most likely encountered a similar situation previously and should therefore be able to answer questions quickly.

It was not surprising to find a significant correlation between programming experience and how quickly students answered each question, Spearman's correlation: $r = -.40$, p (two-tailed) $< .01$. Programming experience seems to express itself in the speed with which code is analyzed for errors.

This analysis only considers the ETH group, since timing information is not available for Birkbeck students.

Hypothesis B: Higher experience results in more correct answers.

Hypothesis A did not consider whether the question was answered correctly or not. It seems likely that an experienced participant will not only answer quickly but also answer correctly.

Again we find a significant correlation between programming experience and number of correct answers, Spearman's correlation: $r = .48$, p (two-tailed) $< .01$ for ETH students.

The result differs for the Birkbeck students: no significant correlation between correctness of answers and programming experience exists. A possible explanation is the homogeneity of programming experience for Birkbeck students. In an attempt to overcome this barrier, if the groups are combined, a significant correlation exists: $r = .48$, p (two-tailed) $< .01$.

Hypothesis C: At a lower experience level, enhanced messages result in more correct answers.

Perhaps the most interesting question is: do novices benefit from messages that provide more information? This would indicate how compilers or tutoring systems for novices can be improved.

Table 3 summarizes the percentage of correct answers obtained for both groups at various experience levels. In the ETH group, the long and short form messages are best for novices. The visual form does not seem to suit them. These results suggest the visual form as one to avoid for beginners.

When establishing the significance of correlations, we find none between the level of experience and the number of correct answers for the long form message. This could be due to the fact

that ETH students are used to the long form message; all students understood it well by the time they answered the questionnaire. For visual inline and short form messages, the level of experience of ETH students correlates significantly with the number of correct answers: Spearman's correlation, r = .43, p (two-tailed) < .01 and r = .33, p (two-tailed) < .05 respectively. This shows that the higher the experience the more correct answers students get for these message types. The more experienced students are likely to have experimented with other compilers and seen other message types. Since experience is likely to have improved their skills anyway, they would do better than the less experienced students.

In the Birkbeck group, at all experience levels, the visual form message is the least suited to help students understand errors. The long and short form messages do not differ significantly as illustrated in Table 3. Spearman's correlation between the level of experience and any of the message types establishes no significant correlations. A plausible explanation for this is again the homogeneity of the group.

Table 3: % correct answers at various experience levels

		Short Form	Visual Form	Long Form
ETH	**Exp. 1 & 2**	84.62	71.79	84.62
	Exp. 3, 4, 5	95.56	94.44	90.00
Birk-beck	**Exp. 1 & 2**	57.89	47.37	56.14
	Exp. 3&4	66.67	60.00	66.67

While we find no significant correlations in all cases for the Birkbeck group; the ETH group shows a significant correlation between two of the message styles and the level of experience. These findings must be taken with caution since we are working with very small numbers (since grouping students by their level of experience results in small groups). The percentages clearly show that hypothesis C does not hold but indicate the visual form does little to help beginners. When both groups are combined, the visual form remains least suited for novices but there is a significant correlation between all message styles and level of experience (r = .40, p (two-tailed) < .01 for short form; r = .29, p (two-tailed) < 0.05 for long form; r = .48, p (two-tailed) < .01 for visual inline form).

Hypothesis D: More information results in more correct answers.

This hypothesis states that the more information a message provides, the more likely a student is to get a correct answer since he can understand better the origin of the problem. A brief message might not give enough information to understand where the problem is.

Table 4 shows the percentage of correct answers per message style (rightmost column). These results contradict hypothesis D: Both ETH and Birkbeck students scored low for Question $Q7_{l\_acc}$, considered the hardest (72.1% and 33.3% respectively) despite being given more help (through a lengthy message).

The results for hypothesis D are not significant in both groups. For all message types, Wilcoxon signed-rank test indicates that there is no significant difference between the means.

Table 4: % correct answers per message style and error type

		Unknown identifier	Wrong number of args.	Private access violation	*Avg.*
ETH	**Short**	95.3	90.7	90.7	*92.23*
	Visual	79.1	88.4	95.3	*87.60*
	Long	93	100	72.1	*88.37*
Birk-beck	**Short**	83.30	41.70	58.30	*61.10*
	Visual	87.50	8.30	54.20	*50.00*
	Long	75.00	66.70	33.30	*58.33*

Hypothesis E: The error type determines the number of correct answers.

Does the error type influence the number of correct answers? Students usually find it more difficult to answer questions involving complex problems. Hypothesis E complements other results in the study by helping to identify whether a certain error type was overly difficult.

For the ETH students, small differences appear in the percentage of correct answers between error types, but they are not significant.

Tangible differences exist for the Birkbeck students: it is clear that most students were able to find the answer for the *Unknown identifier* problems unlike for the other error types. The percentage of correct answers for individual questions contained in the group of questions relating to *Unknown identifier* (first column of Table 4) uncovers few differences between the message types. Wilcoxon signed-rank test confirms there is a significance in the comparison of the number of correct answers for *Unknown identifier* and *Wrong number of arguments:* z = -3.92, p < .01; *Unknown identifier* and *Private access violation*: z = -3.12, p < .01. This confirms that the result was probably due to the easier nature of the problem rather than influence from the message type. Birkbeck students, unlike their ETH peers, confirm this hypothesis.

Hypothesis F: More information in the error message results in shorter response times.

The more information a message provides, the more there is to read, which increases the time it takes to answer. On the other hand, the longer the message, the more information is available to understand what might have gone wrong. This might help the student to get an answer faster.

The analysis shows that this hypothesis does not hold. According to mean times (247.2,166.9, and 223.0 seconds for short, visual and long form respectively), the message type that contains a visual representation results in significantly faster answers. The analysis resulted in significance between the short and visual styles and long and visual styles with: z = -4.86, p< .01 and z = -4.03, p < .01 respectively. The study assumes that participants did spend some time thinking about the problem and their answers rather than clicking through one of the options (multiple-choice questionnaires may produce biased results if participants do not make a genuine effort when answering questions; with open-ended questions, this risk is reduced, but assessing and grading the answers uniformly is more difficult). Again this analysis only considers the ETH group.

6. DISCUSSION

Table 5 summarizes the results obtained for each hypothesis. Many run counter our expectations but provide an indication of where efforts can be placed to efficiently help novices.

Table 5: Summary of results

Hypothesis	ETH	Birk-beck
A: Higher experience results in faster answers	✓	N/A
B: Higher experience results in more correct answers	✓	×
C: At a lower experience level, enhanced messages result in more correct answers	×	×
D: More information results in more correct answers	×	×
E: The error type determines the number of correct answers	×	✓
F: More information in the error message results in shorter response times	×	N/A

Giving a lot of information in an error message does not necessarily help students get the correct answer, as assumed by Hypothesis D. In particular, we expect novices to benefit most from the provision of additional information. The results' lack of support for hypothesis C demonstrates that this is not necessarily true. Adding timing aspects to this analysis shows that the more experienced the students, the faster they can answer questions and the more likely they obtain correct answers. The confirmation of this result shows normality in the student populations; they behave as expected. These timing analyses also help to examine how novices react to being given more information on errors with relation to time. Additional information should cause either an increase or a decrease in response time. An increase might indicate that the student took the time to read the message rather than skipping it. A decrease might indicate that the student read the information, found it helpful and did not spend a long time trying to find out more on what the error is and where it could be elsewhere. Since there is an increase in time only for questions involving the *Unknown identifier* error type, and the response times for questions involving the long form message type are neither highest nor lowest, we have no indication whether the provision of additional information leads to an increase or decrease.

Results obtained for the ETH group and the Birkbeck group differ: usually where one group exhibits strong tendencies, the other tends to display none or little. The high percentage of correct answers for all the questions in the ETH group indicates that ETH students might have been too advanced in the course and found the questions too easy for them. Birkbeck students exhibit more difficulties. The homogeneity of this group however restricts some of the analysis.

It is surprising to find no consistency in the results for message styles: one message style does not obviously help more than others with respect to the difficulty of the error type. The results of the analysis indicate that the message form does not influence the student's performance.

7. CONCLUSIONS AND FUTURE WORK

The aim of this study was to explore whether the form of compiler error messages can help students learning programming, and especially to find out whether novices benefit from additional information in error messages.

The kind of additional information provided in the long form message does not seem to aid message comprehension, or help identify the error faster or better; novices in particular do not seem to reap significant benefits. We have to look elsewhere when deciding what aspects of compiler messages help novices. Similar studies for various characteristics of compiler error messages (such as technicality of the error message, visual representation, messages with examples, etc.) could identify the aspects that have a strong positive influence on novices. With this knowledge, compiler error messages could be tailored for them.

Improvements to the study involve fine-tuning questions to ensure they require careful thought on the student's part. This might involve open-ended questions to reduce the possibility of quick guesses. Adding more questions per error type and more types of errors covering a broader spectrum of complexity should provide more incontrovertible results. Carrying out the survey earlier, while students are still learning will have the advantage of reducing the possibility of skewing the data because the participants are too experienced.

8. ACKNOWLEDGMENTS

Our thanks go to the participants of both studies who volunteered their time. We are grateful to Manuel Oriol and Andreas Pedroni for useful feedback.

9. REFERENCES

[1] T. Flowers, and A.C. Carver, and J. Jackson. Empowering novice programmers with Gauntlet. *Frontiers in Education*, 2004.

[2] J. Jackson, M. Cobb, C. Carver. Identifying top Java errors for novice programmers. *Frontiers in Education*, 2005.

[3] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the 2006 international Workshop on Computing Education Research* (Canterbury, United Kingdom, September 09 - 10, 2006). ICER '06. ACM Press, New York, NY, 73-84.

[4] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg. The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special Issue on Learning and Teaching Object Technology*, 13(4), 2003.

[5] M.H. Ng Cheong Vee, B. Meyer, and K. L. Mannock. Empirical study of novice errors and error paths in object-oriented programming. *7th Annual HEA-ICS conference*, Dublin, Ireland, 29th -31st August, 2006.

[6] M. Satratzemi, and S. Xinogalos, and V. Dagdidelis. An environment for teaching object-oriented programming: ObjectKarel. *Proceedings of The 3rd IEEE International conference on Advanced Learning Technologies (ICALT 03)*, 342-343.

[7] E.R. Sykes and F. Franek. Presenting JECA: A Java Error Correcting Algorithm for the Java Intelligent Tutoring System. *Proceedings of Conference on Advances in Computer Science and Technologies, ACTS 2004*, St. Thomas, US Virgin Islands, November 2004.