Principles of Asking Effective Questions During Student Problem Solving

Kristy Elizabeth Boyer, William Lahti, Robert Phillips*, Michael D. Wallis*, Mladen A. Vouk, James C. Lester

Department of Computer Science, North Carolina State University, Raleigh, NC
*Dual affiliation with Applied Research Associates Inc., Raleigh, NC
{keboyer, wjlahti, rphilli, mdwallis, vouk, lester}@ncsu.edu

ABSTRACT
Using effective teaching practices is a high priority for educators. One important pedagogical skill for computer science instructors is asking effective questions. This paper presents a set of instructional principles for effective question asking during guided problem solving. We illustrate these principles with results from classifying the questions that untrained human tutors asked while working with students solving an introductory programming problem. We contextualize the findings from the question classification study with principles found within the relevant literature. The results highlight ways that instructors can ask questions to 1) facilitate students' comprehension and decomposition of a problem, 2) encourage planning a solution before implementation, 3) promote self-explanations, and 4) reveal gaps or misconceptions in knowledge. These principles can help computer science educators ask more effective questions in a variety of instructional settings.

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: *Computer Science Education*

General Terms
Design, Human factors

Keywords
Computer science education research, instructional discourse, question-asking, tutoring

1. INTRODUCTION
Instructional discourse and dialogue are ubiquitous in computer science education. Channels for this discourse include formal lecture classes [11, 17], office hour help sessions, online message boards [20], email, blogs and Twitter channels. In each of these situations, instructors have the opportunity to increase the efficacy of classroom time and homework exercises by facilitating students' learning. An instructor's choices during instructional discourse and dialogue can profoundly affect the experience of the students [1]. In particular, carefully formulated questions can encourage students to self-explain [6], promote comprehension of requirements and effective planning of solutions [14], and reveal important gaps in student knowledge [22]. Yet, instructors' natural tendencies may not be to pose these questions, possibly because other approaches seem more direct [9, 18].

The objective of the current work is to identify question-asking principles that promote deep student learning through instructional dialogue. We examine the natural question-asking tendencies of untrained instructors and compare these approaches to principles of effective question asking from the relevant cognitive science, psychology, and computer science education literature. The empirical basis for this research is a collection of dialogues between human tutors and novice computing students working to implement the solution to a programming exercise. We have developed a novel question classification scheme and applied it to the questions that occur in the dialogues. The results of the question classification study, when contextualized with the relevant literature, highlight several patterns of "what not to do" for question asking during guided problem solving. The principles of effective question asking described here have direct, practical implications for computer science educators.

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

2. RELATED WORK
Several lines of research have investigated *student* questions in the context of computer science education. For example, pair programming students ask higher-level questions of the instructors than solo programming students [15]. Intelligent agents that monitor collaborative group activity may encourage students to produce higher quality questions [21]. The instructional setting also influences the nature and topic of student questions. For example, students ask different types of questions in online discussions than they do in class lectures [17].

Studying *student* questions can yield valuable insights into how people come to understand computing. In a complementary vein, investigating the impact of *instructor* questions can lead to more effective question-asking practices on the part of educators. Asking effective questions prompts students to engage in valuable

learning behaviors they might not otherwise have undertaken. For example, one important role of questions is to encourage students to self-explain, which has been shown to be highly beneficial for learning [6], a possible source of the effectiveness of Socratic dialogue [12]. With novice computer scientists, asking effective questions during the early phases of planning a solution can support the students' comprehension and decomposition of the problem at hand [14]. Asking targeted, specific questions is useful for revealing knowledge gaps with novices, who are often unable to articulate their questions [22].

Gaining confidence in asking effective questions is an important step toward developing pedagogical expertise. Research in one-on-one tutoring indicates that novice tutors ask far fewer questions than expert tutors [9]. In addition to asking more questions, skilled instructors recognize that the type of question has important implications for student responses. For example, in software engineering, open-ended questions have been found to elicit a wide range of student responses that are valuable in classroom discussion [19].

This paper builds on the related work noted above by examining the types of questions present in a body of tutorial dialogue for introductory computer science. The results illustrate specific ways in which instructors can improve their question-asking approaches to facilitate student learning.

3. METHODOLOGY

One-on-one tutoring is a valuable arena for conducting research on instructional approaches [18]. The empirical results presented in this paper are based on classifying the questions that occurred naturally in tutoring dialogues between untrained tutors and novice computer science students.

3.1 Tutoring Study Participants

The data consist of instructional dialogues that were recorded during the course of two tutoring studies. Participants were 78 students enrolled in a university introductory computer science class who participated in the study in exchange for a small amount of class credit. The 17 tutors were graduate and upper-division undergraduate students, all with research interests in computer science education and varying degrees of experience in peer tutoring. None of the tutors received formal training in tutoring.

3.2 Tutoring Session Format

Students and tutors reported to separate rooms to ensure anonymity and complete capture of the instructional interaction. They interacted through an Eclipse plug-in that facilitates real-time remote collaboration with textual dialogue [3]. The tutor's interface featured a synchronized view of the students' programming window, and the students and tutors engaged in typed dialogue through a textual dialogue pane. All dialogue and programming actions were recorded in a database.

The tutoring studies were conducted during weeks eight and nine of a twelve-week semester. In both studies, the programming exercise involved applying array data structures and *for* loops to solve a programming problem using Java. Additional details about the participants, programming problems, data collection, and learning outcomes for the two studies are reported in [2, 4].

3.3 Data

The 78 tutoring sessions produced a data set of 10,179 textual dialogue messages. Tutors account for 6,558, or 64.4%, of these messages. This proportion of tutor and student dialogue turns is consistent with data sets from other technical domains such as physics [8]. Our previous research classified all the dialogue messages as statements or questions. Of all the tutor dialogue messages, 714 were classified as questions that were on-topic (off-topic questions include "Is it hot in your room?" or "How are you today?") [2, 4]. This paper reports on the further classification of these 714 questions according to a two-level classification scheme that is intended to capture 1) the *instructional goal* that motivated the question, and 2) the realized *question type*.

3.4 Question Classification

Existing literature on question asking has produced several question classification schemes that have been applied to instructional discourse in a variety of domains [10, 16]. We adapted these taxonomies for application to the computer science dialogues by adding several categories that facilitate classifying problem-based and interactive dialogue questions. The iterative process of creating the final question classification scheme proceeded as follows. First, two researchers independently classified the question goals and types in a training subset of the data. Next, all questions that could not be adequately classified with the existing categories were clustered to create new categories. Finally, the revised scheme was tested on a previously unseen subset of the data. This training and refinement loop concluded when the two researchers classified the questions in a training set with an acceptable level of inter-rater reliability [13], indicating that the taxonomy was sufficiently reliable for the data at hand. Table 1 displays the top level in the hierarchy, the instructional goals. Each of these goals can be instantiated as more than one question type. Table 2 displays these types, which constitute the bottom level of the hierarchy.

Table 1. Instructional Goals for Question Asking

Plan: Establish a problem-solving plan. Ascertain what the student wants, prefers, or intends to do.
Ascertain Student's Knowledge: Find out whether the student knows a particular factual or procedural concept.
Hint: Scaffold the student's problem-solving effort.
Repair Communication: Disambiguate or correct a previous statement or question.
Confirm Understanding: Confirm the student understands a previous problem-solving step or previous tutor statement.
Engage Student: Elicit a response from the student either at the beginning of the tutoring session or after prolonged silence.
Remind/Focus: Direct the student's attention toward a previous statement or problem-solving step.

When the classification scheme had been finalized, an inter-annotator agreement study was conducted. For both levels of the classification scheme, one researcher classified the entire set of 714 tutor questions and a second researcher classified a subset of approximately 17% that had not been used during the training and refinement phase. The Kappa statistic for inter-rater agreement was 0.85 for the question goal classification and 0.84, indicating "very good" reliability [13].

4. RESULTS AND DISCUSSION

The results from classifying tutor questions according to the above taxonomy suggest that the untrained tutors engaged in notable examples of "what not to do" in their question-asking approaches. We begin by examining the frequencies of question

types that occurred, and then consider excerpts from the dialogues that illustrate the importance of some question-asking principles.

Table 2. Instructional Question types (* denotes category not found in classification schemes of [10, 16]**)**

Question Type	Example
Assessment*	Do you think we're done?
Backchannel*	Right?
Calculation	What is 13 % 10?
Causal Antecedent	Why are we getting that error?
Causal Consequence	What if the digit is 10?
Clarification*	What do you mean?
Confirmation*	Does that make sense?
Feature/Concept Completion	What do we want to put in digits[0]?
Definition	What does that mean?
Enablement	How are the digits represented as bar codes?
Focus*	See where the array is declared?
Free Creation	What shall we call it?
Free Option	Should the array be in this method or should it be declared up with the other private variables?
Goal Orientation	Did you intend to declare a variable there?
Hint*	We didn't declare it; should we do it now?
Improvement	Can you see what we could do to fix that?
Judgment	Would you prefer to use math or strings?
Justification	Why are we getting that error?
Knowledge*	Have you ever learned about arrays?
Plan	What should we do next?
Procedural	How do we get the i^{th} element?
Quantification	How many times will this loop repeat?
Status*	Do you have any questions?

4.1 Question Type Frequencies

All of the questions from the introductory computer science dialogues were classified according to the question's *goal* and the question *type*.

As depicted in Figure 1, the most common instructional goal was to ascertain the student's knowledge; this goal accounts for just over one-third of the questions. Giving a hint to the student through a question, which is likely an example of an indirect and polite conversational strategy [5], also occurred frequently. This instructional goal accounted for nearly one-fifth of the tutors' questions.

Figure 2 displays the frequency of each question type. *Hint* questions and *procedural* questions are the most common question types, each accounting for nearly one-fifth of the tutors' questions. The second most common questions are *feature/concept completion* and *knowledge* questions. These question types are both intended to gauge student knowledge, with the former being a question directly about the subject matter, while the latter asks whether the student believes he knows or understands a topic.

Some types of questions are known to occur in tutoring from other domains [10, 16], yet these question types are absent from the introductory computer programming dialogues presented here. Some examples of such questions include *composition* questions that ask about the components of an item, *comparison* questions that ask the student to compare items, *example* questions that ask the student to give or interpret an example, and *interpretation* questions that ask for a subjective viewpoint. The absence of these questions may be due in part to the subject matter of the instructional discourse. For example, relatively few introductory computing questions would involve subjective interpretation. On the other hand, some types of questions, such as examples, might be effective in the computing context but were not utilized by the untrained tutors in these studies.

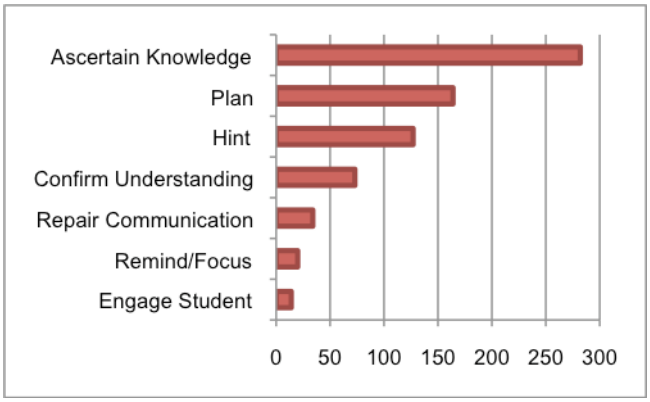

Figure 1. Instructional Goal Frequencies ($n_{questions}$=714)

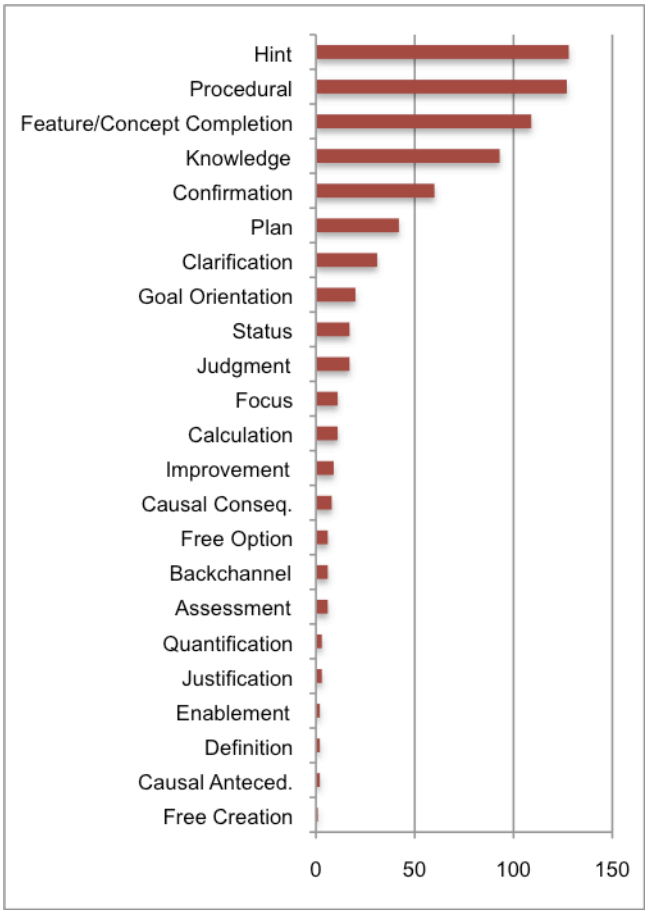

Figure 2. Question Type Frequencies ($n_{questions}$=714)

A small percentage, only 11% of the 6,558 tutor dialogue messages, were questions. This ratio of tutor questions is consistent with findings that untrained instructors, such as the novice tutors used in these studies, tend to ask fewer questions than more highly trained expert instructors [9]. This discrepancy between expert and novice instructors highlights the importance of research on question-asking techniques: effective approaches may not come naturally, yet they hold the potential to improve teaching effectiveness.

The remainder of this section presents excerpts from the data to illustrate the tutors' question asking in context as the student worked to solve the introductory programming problem. For each excerpt, we compare the untrained tutors' question-asking approaches with principles from the literature and discuss how applying these principles may have improved the instructional dialogue.

4.2 Encouraging Comprehension and Decomposition of the Problem

Experienced computer scientists recognize the importance of thoroughly understanding a problem before beginning to construct a solution [14]. In contrast, novices may be more likely to begin implementing a solution without an adequate understanding of the requirements [7]. Figure 3 presents an excerpt that illustrates this phenomenon. The instructional approach illustrated in this excerpt does not encourage the student to think deeply about the problem at hand before beginning to work on the solution; in fact, the tutor encourages the student to move forward with implementation by asking, "Where would you like to start?"

[Tutor and student greet each other]

Tutor 1: Ok, let me know when you're ready to start.

Student 1: I think I'm ready.

Tutor 1: Ok.

Tutor 1: Where would you like to start?

[Goal: Plan, Type: Plan]

Student 1: I guess with the *extractDigits* method.

Tutor 1: Sounds good.

Student 1: I'm already confused.

Figure 3. Not Encouraging Comprehension and Decomposition of the Problem

Because the student had only just finished reading the problem description and had likely not yet fully understood it, questioning the student directly about a plan might have been premature. This possibility is confirmed by what seems like a guess on the part of the student, followed by immediate admission of confusion.

The tutor's goal of establishing a problem-solving plan might have been instantiated more productively as a different question type. For example, a *feature/concept completion* question, (*e.g,* "What are the main steps in completing this assignment?") could have encouraged the student to reflect on and discuss the assignment first. In this context, a useful instructional principle is that *asking specific questions about a problem description can facilitate a student's thorough comprehension of the problem at hand.*

4.3 Eliciting Self-Explanations

At many junctures in instructional discourse, an instructor can choose whether to tell a piece of information to the student or whether to elicit that information by asking a question or a series of questions. Research has shown that self-explaining, in which a student express her own understanding, improves learning [6]. Despite the demonstrated effectiveness of active approaches such as self-explanation, a natural tendency of instructors, especially those with less pedagogical expertise, is to tell the student information that is intended to fill gaps or correct misconceptions in the student's knowledge. Figure 4 illustrates this tendency in an excerpt from the introductory programming dialogues.

Student 2: Ok so *String z = "" + zipcode*?
Tutor 2: Yeah.
Student 2: Then what?
Tutor 2: Ok so now we need somewhere to keep the individual digits.

Figure 4. Tutor "Tells" Rather than Eliciting Student's Explanation

This tutoring session's history has included the tutor giving numerous hints to scaffold the student's problem-solving efforts, and this excerpt represents a transition point at which the tutor could have chosen to ask the student to describe the most reasonable next step (*e.g.*, "Well, given what we've accomplished, what do you think makes sense?"). Instead, the tutor chooses to tell the student what to do next. This illustration of a natural instructional tendency reminds us of the following principle: *Rather than telling the student, asking questions that prompt student self-explanations can improve the student's learning.*

4.4 Asking Targeted Questions

The specificity of questions should be appropriate for the context, which includes the student's knowledge level. Novices often believe they understand when in fact their knowledge is incomplete or incorrect [22]. Figure 5 illustrates an excerpt in which a tutor asks the student whether he has any questions, and the student responds "No." However, after reading a two-page problem description whose solution will constitute his first application of arrays and *for* loops, it is likely that the student has significant knowledge gaps about the problem at hand. The student's lack of experience may render him unable to articulate the questions he has, or he may prefer to begin implementing the solution rather than to discuss his questions.

Tutor 3: Did you have any questions about what the lab wants you to do?

[Goal: Knowledge, Type: Status]

Student 3: No.

Tutor 3: Ok, what method do you want to start with?

[Goal: Plan, Type: Judgment]

Figure 5. Tutor Asks a Vague Question

In this situation, the tutor could have asked the student to explain a key component of the problem description, especially a point that is a known trouble spot for students. For example, an *enablement* question about mechanisms that allow certain behaviors to occur could have been used to illuminate details of the student's understanding (*e.g.*, "How do you think loops will be used to solve this problem?"). A*sking a specific content question can reveal a student's incomplete or incorrect knowledge so that the instruction can proceed productively.*

5. CONCLUSIONS AND FUTURE WORK

Asking effective questions is an important component of pedagogical expertise. By examining tutorial dialogue exchanged between untrained tutors and novice computer science students, we have investigated some types of questions that untrained instructors naturally ask. All of the tutors' questions were classified according to a two-level hierarchical question taxonomy that was based on existing classification schemes enhanced to capture the nuances of the instructional context. The results of this question classification study revealed that the instructors' question-asking approaches could have been improved by applying principles from the relevant literature. We discussed excerpts of tutorial dialogue that lack sophisticated question-asking approaches. The findings illustrate the following principles: 1) *Facilitate comprehension and decomposition.* In problem-solving contexts such as software engineering and programming, asking targeted questions can encourage students to think deeply about the problem at hand. 2) *Prompt for self-explanation.* Periodically asking students to explain their reasoning or understanding results in self-explanation, which has been shown to benefit student learning [6]. 3) *Ask targeted questions.* Content-specific questions, especially questions that focus on known problem areas in the material, are an excellent means for identifying incorrect or incomplete student knowledge. 4) *Ask questions frequently.* Good questions stimulate students to think deeply, explain themselves, and reveal gaps or misconceptions in knowledge. An important component of pedagogical expertise is to ask questions frequently [9].

The work presented here is based on data from tutoring studies. Future work should include experiments designed to assess the effectiveness of specific question-asking approaches. Such experiments would be valuable for confirming the cross-domain applicability of the question-asking principles presented here. Additionally, targeted experiments can shed light on the differences between question-asking approaches for various instructional contexts.

6. ACKNOWLEDGEMENTS

The authors thank Andy Meneely for feedback on this manuscript. This work is supported in part by the NCSU Department of Computer Science along with the National Science Foundation through Grants REC-0632450 and IIS-0812291, a Graduate Research Fellowship, and the STARS Alliance Grant CNS-0540523. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

7. REFERENCES

[1] Barker, L. J. and Garvin-Doxas, K. Making Visible the Behaviors that Influence Learning Environment: A Qualitative Exploration of Computer Science Classrooms. *Computer Science Education*, 14, 2 (2004), 119-145.

[2] Boyer, K. E., Phillips, R., Wallis, M. D., Vouk, M. A. and Lester, J. C. Balancing Cognitive and Motivational Scaffolding in Tutorial Dialogue. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 2008, 239-249.

[3] Boyer, K. E., Dwight, A. A., Fondren, R. T., Vouk, M. A. and Lester, J. C. A Development Environment for Distributed Synchronous Collaborative Programming. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, 2008, 158-162.

[4] Boyer, K. E., Vouk, M. A. and Lester, J. C. The Influence of Learner Characteristics on Task-Oriented Tutorial Dialogue. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, 2007, 365-372.

[5] Brown, P. and Levinson, S. *Politeness: Some Universals in Language Usage*. Cambridge University Press, 1987.

[6] Chi, M. T. H., Leeuw, N., Chiu, M. H. and LaVancher, C. Eliciting Self-Explanations Improves Understanding. *Cognitive Science*, 18, 3 (1994), 439-477.

[7] Ehrlich, K. and Soloway, E. M. An Empirical Investigation of the Tacit Plan Knowledge in Programming. In Thomas, J. and Schneider, M. L. eds. *Human Factors in Computer Systems*. Ablex, 1983.

[8] Forbes-Riley, K., Litman, D., Huettner, A. and Ward, A. Dialogue-Learning Correlations in Spoken Dialogue Tutoring. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, 2005, 225-232.

[9] Glass, M., Kim, J. H., Evens, M. W., Michael, J. A. and Rovick, A. A. Novice vs. Expert Tutors: A Comparison of Style. In *the 10th Midwest Artificial Intelligence and Cognitive Science Conference*, 1999, 43-49.

[10] Graesser, A. C., McMahen, C. L. and Johnson, B. K. Question Asking and Answering. In Gernsbacher, M. A. ed. *Handbook of Psycholinguistics*. Academic Press, San Diego, CA, 1994, 517-523.

[11] Graesser, A. C. and Person, N. K. Question Asking During Tutoring. *American Educational Research Journal*, 31, 1 (1994), 104.

[12] Jones, J. S. Participatory Teaching Methods in Computer Science. *SIGCSE Bulletin*, 19, 1 (1987), 155-160.

[13] Landis, J. R. and Koch, G. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33, 1 (1977), 159-174.

[14] Lane, H. C. and VanLehn, K. Teaching the Tacit Knowledge of Programming to Novices with Natural Language Tutoring. *Computer Science Education*, 15, 3 (2005), 183-201.

[15] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. and Balik, S. Improving the CS1 Experience with Pair Programming. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 2003, 359-362.

[16] Nielsen, R., Buckingham, J., Knoll, G., Marsh, B. and Palen, L. A Taxonomy of Questions for Question Generation. In *Proceedings of the 1st Workshop on Question Generation*, 2008.

[17] Postner, L. and Stevens, R. What Resources do CS1 Students Use and How do They Use Them? *Computer Science Education*, 15, 3 (2005), 165-182.

[18] Ragonis, N. and Hazzan, O. Tutoring Model for Promoting Teaching Skills of Computer Science Prospective Teachers. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, 2008, 276-280.

[19] Razmov, V. and Anderson, R. Pedagogical Techniques Supported by the use of Student Devices in Teaching Software Engineering. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 344-348.

[20] Sheard, J. Electronic Learning Communities: Strategies for Establishment and Management. *SIGCSE Bulletin*, 36, 3 (2004), 37-41.

[21] Soh, L. K., Jiang, H. and Ansorge, C. Agent-Based Cooperative Learning: A Proof-of-Concept Experiment. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 2004, 368-372.

[22] Tenenberg, J. and Murphy, L. Knowing What I Know: An Investigation of Undergraduate Knowledge and Self-Knowledge of Data Structures. *Computer Science Education*, 15, 4 (2005), 297-315.