Debugging & Traceability

RELATED WORK

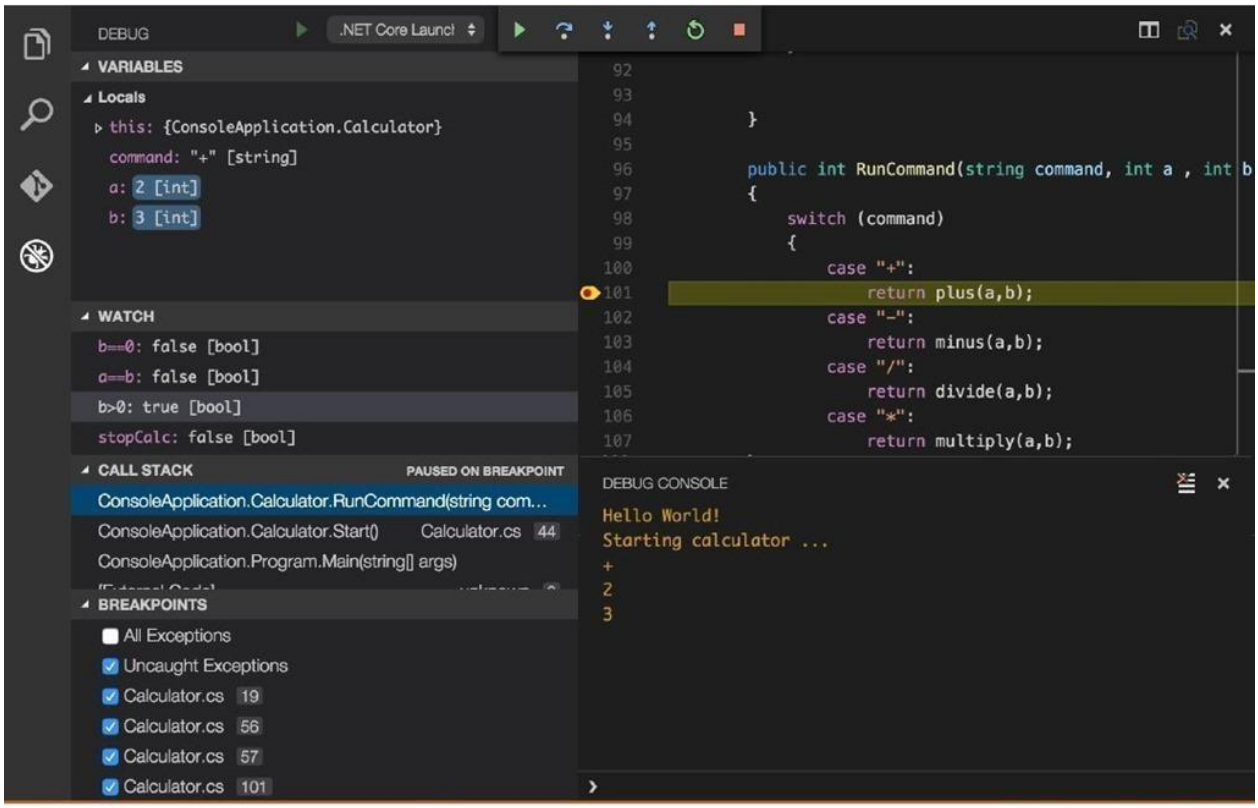
Intelligent tutoring systems (ITSs) often supply a sequence of hints that descend from high- level pointers down to specific, bottom-out hints that spell out exactly how to generate the correct solution. For example, in the Andes Physics Tutoring System, hints were delivered in a sequence: pointing, teaching, and bottom-out [14].

ITSs have been historically expensive and time-consuming to build because they rely heavily on experts to construct hints. Recently, researchers have demonstrated how program synthesis can generate some of the personalized and automatic feedback typically found in ITSs For example, Auto Grader [9] can identify and fix a bug in an incorrect code submission, and then automatically generate sequences of increasingly specific hints about where the bug is and what a student needs to change to fix it.

FORMATIVE STUDY

To understand the current limitations of automatic hint delivery and opportunities to improve it, we observed the hint giving practices of teachers in a local introductory CS course as they helped students debug incorrect code for programming assignments.

We analyzed 132 Q&A posts from the CS course's online discussion forum where instructors answered students' debugging questions. Additionally, we conducted a semi-structured interview with a teaching assistant from the same course to gain insight into the patterns of hint-giving that we observed in the online discussions. This analysis yielded three design guidelines that motivated the design of Trace hint-giving affordances.

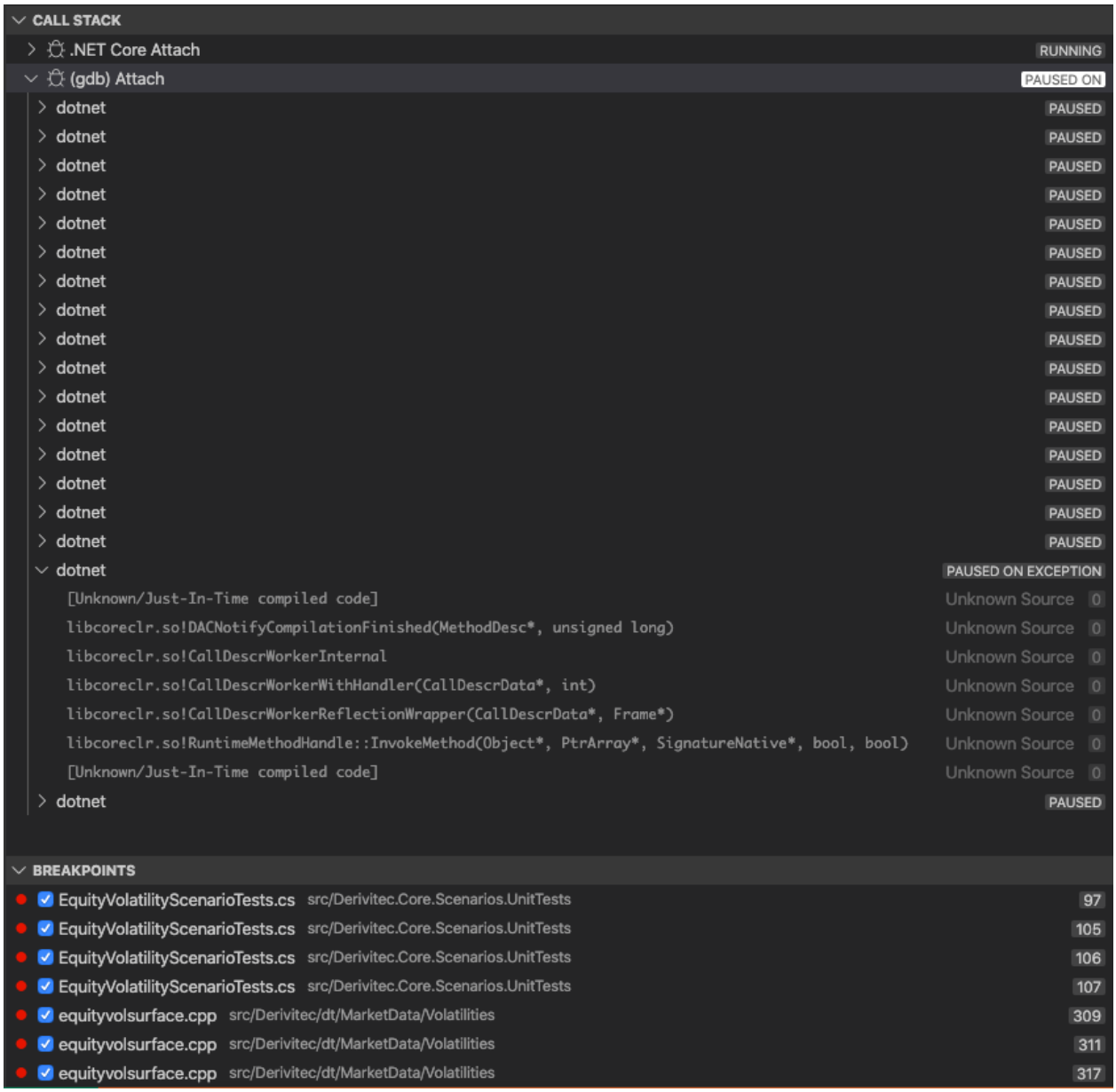


INTERFACE DESIGN

When a student submits an incorrect program to the Trace Diff system, the system back-end synthesizes a fix that corrects the program. This fixed program will be both correct and syntactically close to the student's incorrect submission. The interface can leverage this pair of incorrect and correct programs to show the student the difference between the actual behavior of their submission and expected behavior which would pass all the test cases for the assignment.

The system executes the incorrect and fixed programs, and stores a snapshot of both their internal states at every execution point. Using this information, the system interface, shown in Figure 2, renders execution traces of both the incorrect (D) and fixed (E) programs side-by-side. To help the student find the behavioral differences, the interface highlights where the incorrect program diverges from the fixed one

At the start of each study session, we gave each participant a 6-minute tutorial on both Python Tutor and Trace Diff to familiarize them with each interface. We then gave each participant four incorrect submissions (two for Trace Diff and two for Python Tutor) and asked to perform two tasks for each problem: (1) point out the location of the bug and (2) fix the bug. For each incorrect submission, we explained the programming assignment to the participant and then gave them ten minutes to perform the tasks



EVALUATION

To see if Trace Diff can help students debug their code efficiently, we conducted a controlled experiment and evaluated our interface alongside the Online Python Tutor interface.

We recruited 17 students (male: 15, female: 2; undergraduate: 13, graduate: 4) from a local university to participate in this study. All participants major in computer science and have experience in the Python programming language. In preparation for this study, we collected a dataset of incorrect student submissions to programming problems assigned in CS1, an introductory computer science course at our university.