MAKE INVISIBLE MORE VISIBLE

1. What can you say about source code?

The source code has no innate presence, no innate behavior and does not obey the laws of physics. This is visible when you load it into the editor, but close the editor and it disappears. Think about it a little longer, about how a tree falls, when no one hears it, you begin to wonder if it even existed. A running application has presence and behavior, but shows nothing from the source code from which it was built.

1. What can you say about bugs?

If you’re 90% done and endlessly stuck trying to debug your way through the last 10%, then you’re not 90% done, are you? Fixing bugs is not making progress. You aren’t paid to debug. Debugging is waste. It’s good to make waste more visible so you can see it for what it is and start thinking about trying not to create it in the first place.

1. How to make Invisible visible?

Invisibility can be dangerous. But you can protect yourself from mistakes:

* Using unit tests in each step of your project
* Using bulletin boards and cards makes progress visible and concrete
* Running unit tests provides evidence about the code's behavior. It helps reveal the presence (or absence) of runtime qualities you'd like the appli­cation to exhibit, such as robustness and correctness.

1. What give you visibility?

If your project is apparently on track, and one week later it’s six months late, you have problems—the biggest of which is probably not that it’s six months late, but the invisibility force fields powerful enough to hide six months of lateness! Lack of visible progress is synonymous with lack of progress.

Visibility gives confidence that progress is genuine and not an illusion, deliberate and not unintentional, repeatable and not accidental.

1. What about unit tests?

It is worth paying attention to the start of the project, starting to process the smallest errors, and then you will not have to worry about the invisible consequences. You run the risk of losing more time later on fixing minor issues than if you started debugging at first.

Q/A for article “THE VALUE OF TESTING”

1. What is the value of testing?

Testing allows you to find inaccuracies in calculations, design or engineering errors. Currently testing is mandatory at all stages of development. This reduces the chance of making a small mistake. However, testing is just one of many tools that you can use to improve the quality of code.

1. What can you say about modern programming?

Current programming languages, compilers and static analysis tools are very different. Memory and CPU time are now relatively cheap, so compilers can afford to check for more errors. Almost every language boasts at least one tool that checks for violations of style guides, common mistakes, and sometimes mistakes that are hard to catch, such as potential dereferencing of a null pointer.

1. What kind of tool is lint?

The early C compilers reduced the number of passes through the code that they did by removing some semantic analysis. This meant that the compiler checked only a small subset of errors that could be detected at compile time. To compensate for this, Stephen Johnson wrote a tool called lint that removes errors from your code that implemented some static analyzes that were removed from its sister C compiler.

1. What is a compilator

A compiler is a computer program that translates computer code written in one programming language (source language) into another language (target language). The name compiler is mainly used for programs that translate source code from a high-level programming language to a lower-level language to create an executable program.

1. Should you rely only on testing?

No, don't let testing be your only guarantee of quality. Use analysis tools and don't be afraid to test your own. This is the only way to make truly good software.

Grammar:

26.A  
27.A  
28.B  
29.A  
30.D  
31.D  
32.A  
33.D  
34.D  
35.C  
36.B  
37.B  
38.A  
39.C  
40.B