# Abstract

# Problem Description

Table : - Error Checking

|  |  |  |
| --- | --- | --- |
| Types of Errors | Tools for Essays | Tools for Code |
| Syntactic | Spell Check | Compiler |
| Semantic | Grammar Check |  |
| Substantive | Audience/Grader | User/Tests |

When writing or creating anything from an essay to code, we encounter 3 different types of errors: syntactic, semantic and substantive. To describe each, let us first examine the process of writing an essay in Word. Word has a Spell Checker that immediately alerts the author that their paper contains a syntactic error – specifically that they have written something outside the bounds of the English language. Word also has a Grammar Checker that reports semantic errors – a passage appears to be poorly written inside the English language. Finally one has an audience reading the essay who will highlight any substantive errors – errors within the arguments or premises. When writing code we have a different set of tools. Compilers immediately alert us of semantic errors – our code is not correct. We also have tests or users that can highly substantive errors – the program is not working correctly. However, the area of semantic error checking – whether code is well written – has yet to be automated fully.

Seen from a different point of view, this problem can be formulated in terms of software quality. Namely, there are two perspectives on software quality: that of the user and that of the programmer. Users care about whether or not a piece of software works (aka behaves as it ought to). In contrast, programmers care about how maintainable a piece of software is. Maintainability minimally implies code is easy to read and update. There are ways to evaluate the user’s perspective of a program, most easily through automated testing. Though there are tools to evaluate the programmer’s perspective, these tools only check specific aspects of code. Given that code quality is subjective, any tool that only performs pre-defined inspections will never be satisfactory to every programmer.

While I have talked about the nature of semantic error checking, I have yet to comment on why it is important. The biggest reason to perform semantic error checking is to improve readability or the ease with which another programmer can understand the code. In the same way that grammatical errors in a paper can confound its underlying arguments, poorly written code can easily obscure its underlying function. In an industry or academic setting where other individuals will necessarily need to read one’s code, readability is at a premium. Similarly, readable code is easy to revise and update later. In an academic environment, automated semantic error checking immediately helps save work for professors and TAs by producing automated comments instead of needing to write the same set of stylistic comments to multiple students. Students can also directly benefit by applying this tool to their code before submitting assignments; giving them the chance to improve their grades and their coding habits.

# Related Products

## Functionally

Java (PMD, Checkstyle) and Splint

## Design

Frama-C

## Java options

<http://pmd.sourceforge.net/>

<http://checkstyle.sourceforge.net/index.html>

## Splint

### Email from Splint people

## BLAST

## Frama C

## Clang

## Plugin Architecture

### Eclipse

### YouTube