CE 316 - Project Description

The Integrated Assignment Environment

Large projects require different technologies to work together. For example, a compiler uses several other tools until it creates an executable file. Similarly, an integrated development environment (IDE), such as IntelliJ, Eclipse, or VS Code, uses the compiler, a Git client, syntax highlighting, and many other tools to provide a better programming experience.

Your project is a lightweight integrated assignment environment (IAE) for managing programming assignments. The lecturer must be able to create a new assignment by setting its environment. For example, let's assume that the assignment is about a C programming task. The user must be able to provide the steps to (i) compile a C program, (ii) run the executable with parameters, (iii) check if the output is correct, and (iv) report the status of each student's assignment.

There are several challenges here; some programming languages are compiled (such as C, C++ and Java) and some are interpreted (such as Python). So the steps might be different for different scenarios. The user must be able to create these scenarios as configurations, and then use them in assignments that will be created later. The user will probably share these configurations with others, so they must be on separate files. These files would also help if the software is installed to a new computer. Existing configurations could be used without having to create them again.

The arguments to the executable, generally referred as "command line arguments" or "arguments to main" are important because it is possible that the lecturer will provide an input which the students did not see before to check the correctness of the program. For example, let's say that the assignment is about sorting. Then the user must be able to provide an input and an expected output, similar to a JUnit test, so that all student programs could be checked quickly.

The hardest part is to include the files submitted by the students. It could be assumed that these will be ZIP files, and when they are extracted they will be in a new directory with the student ID.

Finally, the IAE must be robust. If there are errors in student files, such as a compilation error, it should be reported and the software must continue with the next student.

Here is a full scenario:

> The lecturer prepares an assignment; a string sorting problem which must be developed in C. The list of strings must be passed to the main via the command line arguments. The assignment must be saved as a main.c file.
>
> Once the assignment is given, the lecturer creates a new "project" on the IAE. The project will be using the "C Programming Language Configuration." This configuration is either

created earlier, or the lecturer creates it. The configuration says that the IAE should use the /usr/bin/gcc command to compile the source code “main.c” and should use “-o main” as a parameter to the compiler so that the output is saved as main. This is step (i).

In step (ii) the parameters to the executable must be provided. This could be done either by entering the strings manually or by creating a program that generates them. This is up to the lecturer. Typically, the lecturer will have a file that contains the strings and another file that contains the sorted version. So the strings in that file could be passed as command line arguments to “main” or a command line script could be used to generate them. The IAE should provide this.

In the next step, step (iii), the output must be compared to the sorted version of the strings. There are command line applications that can compare these two files, or the IAE could do it, or another program could be developed to do this. The result of this comparison will be stored in a report either as success or failure.

The final step (iv) shows the results of all students.

Once the lecturer prepares the IAE he/she downloads assignments in ZIP format and places them in a directory. He/She points the IAE to the directory. Assuming that the students were able to zip the files correctly, a directory with the student ID is created when each ZIP file is unzipped. Inside each directory there must be a main.c file which will be compiled. When the lecturer points to the directory and hits “run” on the IAE, the IAE will process each ZIP file according to the steps (i) to (iv) and will report the results. These reports must be saved within the project itself along with any errors that the IAE encounters.

The lecturer must be able to open any project he has created earlier and see the results.

The End!

Please notice that you need to be able to run several programs within the IAE. During compilation for a C program for example you have to be able to call gcc within the IAE and check if it returns successfully or not. You can use existing programs to compare the program output to the expected output. You can also use an unzip command to decompress the files submitted by users. The software will operate on files submitted by several students, so there will be a loop of these steps (compile, run with arguments, check the output) and the results must be reported. These reports must be saved and viewed within the IAE.

Since this is a large project, the design must be carefully done. Please design first. It is advised that the IAE would be a GUI program calling other programs that either exist in the system (what if they don’t exist!) or they are developed by you!

This project must have an installer and must not depend on any servers. If the software uses any library, make sure the required files are installed while the software is installed. The software cannot depend on a server, such as MySQL or any other server. It must be able to stand on its own. You can assume that the compilers and interpreters exist on the lecturer's computer and are ready to run from the command line on the lecturer's computer, but you

cannot assume that they have “zip” or “unzip” on the command line. The same can be said for other tools, such as checking if the output and the answer are the same.

Here are the requirements. Your software is expected to satisfy them all.

Requirements

Requirement 1 (system requirement): The software must be deployed to target Windows computers with an installer.

Rationale and Remarks: While I’m aware that some of you use Linux or MacOS, we should stick to Windows for this case. I cannot test all projects on different computers, so the target platform is Windows. The software must have an installer and this installer should deploy all the required executable files, static files, and libraries to the target system. You can assume that the target system has Java installed, but not JavaFX - so check out JavaFX documentation to find out how to distribute an application that uses JavaFX. The installer must add a shortcut icon to the desktop, so that I can start it from there. Check out Windows Installer or Inno Setup (https://jrsoftware.org/isinfo.php) for installers.

Requirement 2 (system requirement): The software must have a manual that will be displayed with a “Help” menu item.

Rationale and Remarks: All software applications come with their documentation. This is no exception. There must be a manual on how to use the software and this manual must be accessible from the “Help” menu on your application.

Requirement 3 (user requirement): The user must be able to create a project that uses an existing or a new configuration.

Rationale and Remarks: This is the core functionality. The project holds together all related information: the configuration, the results, the submitted student files, etc. The project information must be kept on a file or on a file-based SQL provider, such as SQLite. You cannot assume that the lecturer will install and configure a SQL server for you, so please use SQLite or another file based SQL implementation if you choose to use SQL.

Requirement 4 (user requirement): The user must be able to create, edit and remove a configuration.

Rationale and Remarks: A configuration is how a programming language is handled. The way we compile and run a C program is different from a Java program. Therefore, the compiler path and parameters, how the program is run, and all related settings must be stated here.

Requirement 5 (user requirement): The user must be able to import and export configurations.

Rationale and Remarks: This is an important requirement: it basically says that the user can create back ups of these configurations so that they don't have to create them when the software is installed to another computer. It also gives an opportunity to share them with others. Notice that if you use SQL then you must provide a way to import and export these files to the SQL database.

Requirement 6 (user requirement): The software must be able to process ZIP files for each student.

Rationale and Remarks: ZIP is a very common format. All operating systems have built-in support for it. The emphasis here is on the multiple files that the software must operate on. The software must be able to process a list of ZIP files in a directory. This MUST NOT be done manually, one by one.

Requirement 7 (user requirement): The software must be able to compile or interpret source code using the configuration of the project.

Rationale and Remarks: This supports the core functionality of the software. The software must be able to use the configuration file and must be able to compile (or interpret) source code after unzipping the files.

Requirement 8 (user requirement): The software must be able to compare the output of the student program and the expected output.

Rationale and Remarks: After compiling and running the student source code, the software must be able to compare the output of the program to an existing output to decide if the source code operates as expected or not.

Requirement 9 (user requirement): The software must be able to display the results of each student file.

Rationale and Remarks: Once the software runs all student programs and gets their results, it must be able to show the results.

Requirement 10 (user requirement): The user must be able to open and save projects to operate on them at any time.

Rationale and Remarks: The user must be able to see the results of a project at a later time. This is an expected behavior where the user can see the status of a project simply by opening the project.

Design Document

A design document basically describes how you will handle all the requirements in the previous section. It is possible to implement the software just by looking at this document. The software must be **designed** - that is you have to decide the classes, their relations, and in some cases, the algorithms to use. Then, this design must be documented. To do so, we use text that is supported by UML. It does not make any sense if you only provide the UML.

There are some standards available, especially IEEE templates for the design document. However, they are very comprehensive. For this course, the requirement is only to communicate your design. **The main question is "Can someone build our software just by looking at this document?" - If your answer is "Yes", then your design document is appropriate for this project.**