XXXIV Неделя науки СПбГПУ. Материалы межвузовской научно-технической конференции.
Ч.XI: С.18-19, 2006.

УДК 681.326(075)

И.К.Морев (2 курс, ФТК), Д.М.Гензелев (2 курс, ФТК),
А.О.Рогозов (2 курс, ФТК), Т.А.Быстрова, доц.

АВТОМАТИЗАЦИЯ ПРОЦЕССА ТЕСТИРОВАНИЯ СТУДЕНТОВ В ОБЛАСТИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Экзамены и различные тесты являются важными ступенями в процессе обучения. Таким образом, специальное программное обеспечение для проверки знаний и навыков студентов может быть очень полезным. Процесс проверки студентов в написании программ и различных алгоритмов на разных языках программирования можно свести к анализу скорости выполнения, занимаемой памяти и результатов вычислений. Если проверка последнего требования может быть осуществлена вручную, то проследить за выделением ресурсов и скоростью работы программы преподавателю практически невозможно. В случае большого количества заданий этот процесс еще больше усложняется, и мы приходим к выводу, что для качественного тестирования необходима его полная автоматизация. С точки зрения преподавателя, данный подход поможет ему в полной мере проверить результаты работы выполненных студентами программ и, несомненно, ускорит процесс проведения экзамена.

Examinations and different tests are very important steps in the education process. So special software for students testing may be very useful. Usually the main part of student's testing process is connected with checking the implementation of such requirements as the correctness of the output file, the speed of the equations and the amount of memory needed for program execution. It's obvious that manual analysis of the results is rather difficult, especially in the case of large quantity of the tasks. During programming competitions the quantity of students also makes manual tests impossible. So the only solution is making the testing process fully automatic. From the instructor's point of view this will make the results more accurate and will speed up student's examination. From the student's point of view this will give opportunity to run tests without teacher's help.
T3 system is the only system that works with executables and defends the OS and itself from dangerous code. Usually such applications make syntax analysis of the source file and if it contains no "bad" functions it is compiled and executed. Such approach makes a lot of limitations. First of all, the number of programming languages available for writing tasks is limited with the compilers quantity. The second problem is connected with searching for bad code or functions. There are a lot of cases when the source has no bad functions, but due to "assertion" checks, LoadLibrary calls and so on, the compiled source causes the error.
We suggest another approach. Our method is based on creating the so called Safe Area. This concept includes several types of protection. First of all we create special system user account. It has rather low privileges and rights. All user tasks a started under this safe account. So the executed program has access only to safe directory. If the user tries to output some data to the file located somewhere outside the safe directory, this task will be automatically stopped.
As all the students should have the same run-time conditions and all the data related to the tasks should be stored at the single place we use Client-Server approach. The server requirements are rather high. But from the other hand we get resource undemanding client application. It can run under Windows 98 without any troubles.
The communication between the client and the server is built upon internal protocol based on TCP protocol. We use asynchronous windows sockets to transfer data from client to server and back. The

T3 protocol allows receiving and sending data or files and getting response to all this operations. And every single transaction is made with security checks.
For obtaining data from the database we use ADO .NET technology. Though the T3 Server uses Native Win32 functions, we have chosen this technology for simplicity and data provider independence. The only problem is transferring objects between Native and .NET (managed) code. But we've solved it using COM Callable Wrapper.