УДК 372.016:004

ПРОФИЛИ И ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ В СИСТЕМЕ ТЕСТИРОВАНИЯ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ BM TESTING

***В.Э. Бойко, Ю. Мелеш**, 3 курс*
*Научный руководитель – **Н.В. Силаев**, доцент*
Брестский Государственный Университет имени А.С. Пушкина

Система тестирования задач по программированию BM Testing — это комплекс программ, направленный на проверку практических навыков студентов в области программирования. Невозможно достаточно подробно описать всю эту систему за один раз, по этому, здесь описывается только использование профилей в системе и применение переменных окружения при настройке сервера.

Идеи профилей применяется почти во всех серьезных приложениях, т.к. на основе их реализовывается многопользовательское их использование. Наша система также позволяет создавать профили. Профили выгодны, когда система, например, применяется несколькими преподавателями или на нескольких разных дисциплинах. Использование профилей позволяет разделить данные, что позволяет избежать большого накопления данных в одном месте и не дает запутаться в них. Все настройки связанные с профилями также хранятся в реестре, где для каждого из них создается отдельная ветвь. Все это хорошо скрыто от администратора, что облегчает его работу. Для

каждого профиля создаются отдельные базы данных. Какую из них использовать, можно вручную настраивать, но в большинстве случаев система все это сделает за вас, что, опять же, упрощает настройки и экономит время администратора. Также на сервере реализована возможность импорта данных из других профилей, что избавляет администратора от многократного ввода одних и тех же данных и опять же сильно экономит его время. На создание профиля также уходит не более нескольких минут. Почти всё, как сказано выше, сервер может настроить автоматически(это рекомендуется). Использование профилей помогает также в ситуациях, когда некоторые мероприятия проводятся несколько раз(допустим коллоквиумы или экзамены). С использованием профилей нет смысла создавать эти мероприятия заново, достаточно создать под них профиль и при надобности просто их загружать, уже не думая о задачах, студентах и компиляторах, которые необходимо добавить в базы данных. Все это очень упрощает работу с программой, позволяет работать оперативно, избавляет от рутинных и монотонных действий.

Из сказанного выше можно подумать, что использование множества различных профилей, может привести к сильному увеличению размера программы, однако этого не происходит. Давайте разберемся почему. Опять же все связано с самим движком программы, который изначально задумывался для создания универсального приложения, которое будет рационально расходовать ресурсы. При добавлении какой-то задачи к профилю, при условии, что она уже существует где-то в другом профиле, она не копируется туда полностью, а система просто создает ссылку на нее, тем самым мы сильно уменьшаем размеры программы, увеличиваем производительность системы и повышаем мобильность всего комплекса.

Добавление компиляторов очень сложная процедура вообще в целом, по этому в нашей системе можно использовать переменные окружения, которые хранят некоторую нужную информацию. Например, переменная хранящая директорию тестирования задачи называется «TESTING_DIR», а переменная, хранящая имя файла задачи, присланной от клиента, называется «TASK_NAME». Для того, чтобы Parser системы смог понять, что это переменная, а не просто текст нужно обрамлять ее знаком «%». Например, строка компиляции для компилятора Microsoft Visual C# будет выглядеть так:

```
csc/out:%TESTING_DIR%%TASK_NAME%.exe %TESTING_DIR%%TASK_NAME%.cs
```

Из примера видно, как переменные окружения хорошо помогают в случаях, когда некоторые данные заранее не известны.

В системе имеется возможность использовать и настраивать процедуру загрузки. Используя процедуру загрузки можно, оградить администратора от монотонных действий, при запуске системы, например, запуска сервера или смены режима работы. В процедуре загрузке, каждая команда должна писаться с новой строки, между командами может быть одна или несколько пустых строк. Все, что будет написано кроме известных Parser команд, будет им проигнорировано. Parser понимает, например, команды: TYPE_E, TYPE_L SERVER_ON, SERVER_OFF. Вот простой пример, который запускает сервер и включает лабораторный режим работы:

```
TYPE_L
SERVER_ON
```

Как видно из статьи, комплекс BM Testing является очень гибким в отношении настроек. Продуманная система управления серверной части дает возможность настроить систему для различных ситуаций. Однако, не смотря на все преимущества данного подхода, остается пока много нерешенных проблем. В ближайшем будущем планируется создание новой системы. Используя опыт, полученный при разработке системы тестирования BM Testing, мы будем пытаться создать среду, которая будет более мобильной и еще более продуктивной.

В настоящее время использование профилей и переменных окружения в системе тестирования задач по программированию BM Testing сильно облегчают работу администратора данной системы на математическом факультете Брестского Государственного Университета имени Пушкина.