АВТОМАТИЗИРОВАННАЯ СРЕДА ПРОВЕРКИ ПРАКТИЧЕСКИХ ЗАДАНИЙ, СОДЕРЖАЩИХ РАЗРАБОТКУ ПРОГРАММНЫХ СРЕДСТВ

Д.С. Мойса, Л.В. Серебряная

Белорусский государственный университет информатики и радиоэлектроники, Минск, Беларусь, dmitrymoisa@gmail.com, l_silver@mail.ru

Abstract. It is important to involve students of IT specialties into the real teamwork processes at early stages of higher education to make the future involvement at their real workplaces easier. An environment for automating the processes of checking and evaluating practical tasks, which helps students to study real teamwork methods and tools is proposed.

В настоящее время дистанционная форма обучения стала очень популярной. Однако дистанционное взаимодействия тьютора и студента порождает сложности в организации проверки работ по предметам, предполагающим написание программных средств. Часто возникают сложности с работоспособностью этих программ в различных окружениях. Студент может написать программу на своем домашнем компьютере, отправить ее тьютору, однако на машине преподавателя лабораторная работа может не запускается. Для этого может быть множество причин: отсутствие необходимых библиотек или определенного ПО, не полностью настроенное окружение, отсутствие лицензий, и т. д [1]. Кроме того, для студентов специальностей в сфере информационных технологий основополагающее значение имеет глубокое изучение всех этапов процессов разработки и сопровождения программных средств. Очень важно, чтобы студент, как можно раньше мог изучить и, самое важное, участвовать и набираться опыта в процессах командной разработки программного обеспечения, близких к реальным технологиям, использующимся в коммерческой разработке. Это позволит ему в будущем быстрее вливаться в рабочий процесс на рабочем месте после окончания университета.

В данной работе рассматривается техническое обеспечение и организация процесса выполнения и сдачи практических работ, предполагающих написание программных средств. Предлагаемое решение помогает решить упомянутые выше проблемы. Прежде всего, такая организация системы построения, тестирования и конфигурирования программных средств основана на практиках, применяемых в коммерческой разработке ПО. Практики непрерывной поставки и контейнеризации хорошо зарекомендовали себя и при внедрении в процесс дистанционного образования помогут обеспечить единообразность процессов сборки, тестирования и функционирования на машинах студентов и тьюторов и исключить фактор влияния отличия сред на результаты работы [2]. Кроме того, такая организация процесса позволит вовлечь студентов в реальные процессы разработки с самых ранних стадий обучения. Выполняя работы по описанному процессу, студент получит такие важные для дальнейшей работы знания и навыки, как работа с системами контроля версий, серверами непрерывной интеграции, контейнеризации, изучит процессы управления конфигурациями, практики непрерывной интеграции и развертывания ПО.

В состав среды входят такие компоненты, как система контроля версий, сервер непрерывной интеграции, хранилище образов (в его роли может выступать ПС для хранения бинарных файлов), а также сервера, куда в результате происходит разворачивание готовых контейнеров для проверки. Схема среды организации процесса изображена на рисунке 1.

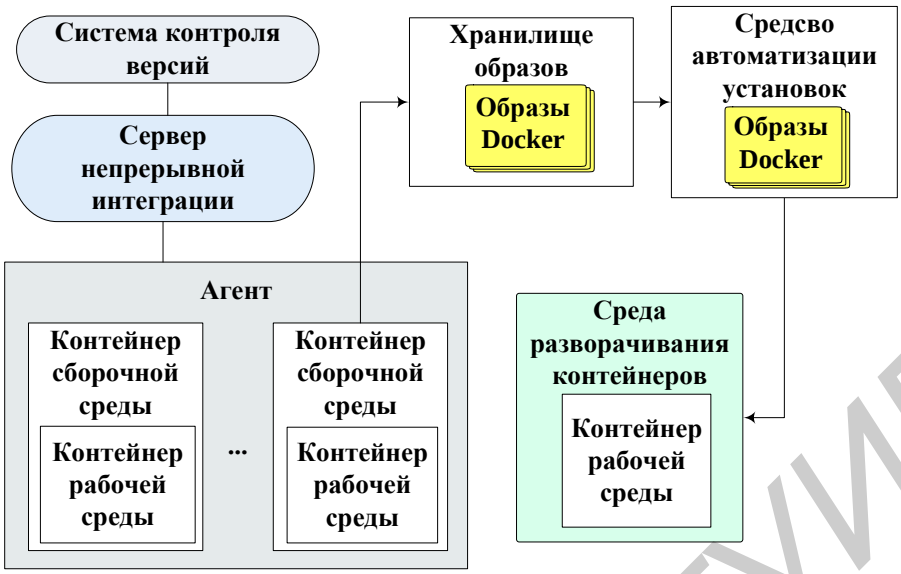

Рисунок 1 – Схема среды для организации проверки лабораторных работ

Процесс выполнения и приема лабораторной работы в данном случае выглядит следующим образом:

1. Студент получает задание. При этом тьютор создает для него репозиторий в системе контроля версий и дает к нему доступ.
2. Студент описывает контейнер (окружение), в котором работает его программа.
3. Студент выполняет лабораторную работу, проверяя работу программного средства внутри контейнера.
4. При попадании работы в систему контроля версий сервер непрерывной интеграции производит автоматическое разворачивание контейнера из репозитория. При этом возможен запуск статических анализаторов кода, результаты его работы могут быть использованы при оценке работы.
5. Когда студент решает, что работа готова, он, с помощью сервера непрерывной интеграции отправляет работу (образ контейнера) в хранилище образов.
6. Преподаватель получает уведомление, что работа готова к проверке.
7. Преподаватель инициирует разворачивание контейнера из образа на сервер. В результате он получает ту же среду, которую использовал студент.
8. Преподаватель проверяет работу на соответствие заданию.

Таким образом, при внедрении предложенного процесса выполнения и приема практических работ, работоспособность программных средств, написанных в рамках этих работ, не зависит от окружения, в котором работа проверяется, так как оно полностью повторяет окружение, которое использовал студент при выполнении работы. Кроме того, уже с самых начальных стадий обучения студент вовлекается в использование инструментов и методов командной разработки ПО, что может помочь ему в будущей профессиональной деятельности.

Литература

1. Turnbull James – The Docker Book. Containerization is the new virtualization. - dockerbook.com, 2014

2. Humble, J. Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation – Boston: Addison Wesley, 2011