ИНСТРУМЕНТЫ ДЛЯ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ ПРЕПОДАВАТЕЛЕЙ И СТУДЕНТОВ С ИСПОЛЬЗОВАНИЕМ СИСТЕМ КОНТРОЛЯ ВЕРСИЙ

Ю. А. Протасевич[1], **О. А. Змеев**[1], **Д. А. Соколов**[1]

[1] ***Национальный исследовательский Томский государственный университет***
634050, Россия, г. Томск, пр-т Ленина, д. 36

Аннотация

В статье описывается подход к организации взаимодействия преподавателя и студентов на курсах по программированию при использовании системы контроля версий Git. Производится сравнительный анализ различных систем управления Git-репозиториями с целью определения наиболее подходящей для образовательных нужд и доступной системы. На основе опыта различных учебных учреждений, применяющих системы контроля версий на своих курсах, были определены положительные стороны и недостатки использования данных систем при обучении. Учитывая существующие проблемы, было разработано решение на основе системы GitLab. В рамках этого решения предложен способ организации работы преподавателя и студентов в дисциплинах, в которых применяются системы контроля версий. Данный подход предполагает использование GitLab в совокупности с разработанной системой, которая является менеджером по управлению Git-репозиториями и предназначена облегчить работу преподавателя и администратора, автоматизируя выполняемые ими задачи. Основной целью статьи является подробное описание этого подхода: системы ограничения прав участников образовательного процесса, организация и функциональные возможности GitLab, список решаемых задач для каждого участника. Также в статье описана организация работы в созданной системе, показаны ее основные сущности и их взаимосвязи, приведен обзор возможностей, которые предоставляет система.

Ключевые слова: автоматизация процесса обучения, системы контроля версий, Git, GitLab.

DOI: 10.32517/0234-0453-2021-36-4-36-46

Для цитирования:

Протасевич Ю. А., Змеев О. А., Соколов Д. А. Инструменты для организации взаимодействия преподавателей и студентов с использованием систем контроля версий // Информатика и образование. 2021. № 4. С. 36–46.

Статья поступила в редакцию: 31 января 2021 года.

Статья принята к печати: 6 апреля 2021 года.

Сведения об авторах

Протасевич Юлия Алексеевна, магистрант кафедры программной инженерии, Институт прикладной математики и компьютерных наук, Национальный исследовательский Томский государственный университет, г. Томск, Россия; protasevich.yuliya@gmail.com; ORCID: 0000-0002-0054-9138

Змеев Олег Алексеевич, доктор физ.-мат. наук, профессор, профессор кафедры программной инженерии, Институт прикладной математики и компьютерных наук, Национальный исследовательский Томский государственный университет, г. Томск, Россия; ozmeyev@gmail.com; ORCID: 0000-0002-8170-4290

Соколов Данила Александрович, начальник управления цифровых решений, Национальный исследовательский Томский государственный университет, г. Томск, Россия; danila.sokolov@accounts.tsu.ru; ORCID: 0000-0001-7992-1205

1. Введение

Организация эффективного взаимодействия между преподавателями и студентами — одна из важнейших задач вуза. Правильно выстроенное взаимодействие положительно сказывается на производительности всех участников, вовлеченных в учебный процесс: позволяет эффективно распределить время и трудозатраты обеих сторон, повысить качество их работы, сделать образовательный процесс более познавательным и увлекательным. Достичь поставленной задачи помогает внедрение различных автоматизированных систем поддержки обучения. Одной из таких систем является система контроля версий, которая может использоваться на дисциплинах по программированию.

Во время проведения занятий по таким дисциплинам преподавателю необходимо выполнять такие задачи, как публикация материалов для студентов, отправка заданий, просмотр и комментирование исходного кода выполненных работ, отслеживание прогресса обучающихся. При наличии большого количества студентов на дисциплине раздача и проверка их работ сильно усложняются, и поддержка такой дисциплины становится трудоемкой задачей. У студентов в свою очередь возникает необходимость долговременного хранения исходного кода программ, синхронизации и отслеживания внесенных изменений при командной разработке [1].

Функциональность, необходимую для решения вышеперечисленных задач, предоставляют ***системы контроля версий*** (СКВ) — *системы, которые позволяют хранить несколько версий одного и того же документа и предоставляют инструменты для управления этими версиями: ведение журнала изменений (кто и когда вносил изменения), разрешение конфликтов между версиями, возможность возврата к предыдущим версиям*. СКВ являются необходимым инструментом при совместной работе над проектом нескольких разработчиков и широко применяются в профессиональной разработке.

СКВ отлично подходят для организации взаимодействия между преподавателем и студентами

в тех дисциплинах, где обучающиеся выполняют лабораторные задания по программированию, работая над проектами самостоятельно или в команде. СКВ позволяют вести учет изменений, которые вносились в проект, что облегчает проверку работ для преподавателя, делая возможным отслеживание производительности студента, долю его вклада в командный проект. Студенты при таком подходе всегда имеют надежное хранилище своего кода, могут вести разработку в команде, без труда объединяя внесенные изменения каждого члена команды, имеют возможность получить полезный навык работы с инструментом, который повсеместно используется в профессиональной разработке.

2. Git и системы управления репозиториями

Для организации взаимодействия между преподавателем и студентами в качестве системы контроля версий нами был выбран **Git** [2, 3] в силу своей популярности, удобства и предоставляемого функционала. Git позволяет хранить каждую новую версию проекта и предоставляет мощные инструменты для управления версиями. Git относится к классу распределенных СКВ со следующей моделью работы: у каждого пользователя есть своя версия репозитория, которая хранится в локальном хранилище, есть возможность добавлять и забирать изменения из любого репозитория. При этом существует условный *центральный репозиторий*, в который разработчики отправляют изменения из локальных репозиториев и с которым они синхронизируют свои локальные репозитории. После внесения достаточного количества изменений в локальную копию эти изменения отправляются в единое хранилище на сервер.

Однако Git сам по себе не предоставляет решение для *управления* репозиториями — места (хранилища), где хранятся все файлы вместе с историей их изменения и другой служебной информацией. Для организации такого управления необходимо использовать сторонний инструмент. Примерами таких инструментов являются GitHub [4], GitLab [5], BitBucket [6] и др.

Ниже представлена таблица, в которой сравниваются различные системы управления репозиториями по удобству и доступности их применения в образовательных целях.

Таблица 1

Сравнение возможностей систем управления репозиториями

Возможности	GitLab	GitHub	BitBucket
SAAS	+	+	+
Неограниченные публичные репозитории	+	+	+
Неограниченные приватные репозитории	+	+	+
Трекер задач	+	+	+
API	+	+	+
Бесплатная версия	+	+	до 5 польз.
Полная версия (за 100 пользователей)	$400/мес.	$400/мес.	$600/мес.
Полная версия (за 100 пользователей) для учебных заведений	$0	$0	$0
Self-Hosted	+	+	+
Неограниченные публичные репозитории	+	+	+
Неограниченные приватные репозитории	+	+	+
Трекер задач	+	+	+
API	+	+	+
Бесплатная версия	+	–	–
Полная версия (за 100 пользователей)	$4800/год	$25200/год	$9500/год
Полная версия (за 100 пользователей) для учебных заведений	$0	$0	$4750/год
Развертывание на Linux	+	–	+
Развертывание на Windows	–	–	до 500 польз.
Развертывание в виде Docker-контейнера	+	–	+
Развертывание в виде готовой виртуальной машины	–	+	–

Среди рассмотренных систем управления репозиториями самым доступным для образовательных учреждений является **GitLab**, имеющий бесплатную версию, которую можно развернуть на своем сервере [7] без ограничения на количество пользователей и которая не требует подтверждения аккредитации.

3. Проблемы при использовании СКВ

Использование СКВ в образовательном процессе имеет много плюсов, облегчает и делает обучение программированию более эффективным [8–11].

Однако при использовании СКВ возникает ряд проблем и сложностей, которые могут затруднить ее внедрение. Данные проблемы упоминаются многими авторами, использовавшими СКВ для проведения своих курсов по программированию.

Ниже представлен **список тех проблем в работе с СКВ, которые были выявлены разными исследователями данного вопроса:**

- отсутствие опыта работы с СКВ и системами управления ими как у студентов, так и у преподавателей [12, 13]. При наличии достаточно высокого доступа в системе пользователи могут намеренно или случайно повредить репозитории [12, 14], совершив следующие действия:
  - сделать опечатку в названии репозитория, что затруднит использование средств автоматизации, которые могут применяться для сбора заданий и вычисления активности студентов;
  - удалить репозиторий с заданием, потеряв все свои и/или чужие изменения в нем;
  - удалить отдельные коммиты* из репозитория путем выполнения команды: `git push -f`;
  - предоставить доступ к своему репозиторию третьим лицам;
  - не предоставить преподавателю доступ к нужному репозиторию либо лишить его этого доступа;
  - переписать историю журнала таким образом, чтобы время выполнения заданий изменилось на более раннее по сравнению с тем, когда было выполнено в действительности;
- студенты видят репозитории друг друга и могут заимствовать решения [12];
- сложность понимания работы с Git на начальных этапах работы [13, 15];
- сложность начальной установки, конфигурации и администрирования СКВ [14];
- проблематичность поддержания курсов с большим количеством пользователей, репозиториев, заданий [14, 15].

Авторы также отмечают, что СКВ больше ориентированы на использование в профессиональной разработке, нежели в образовании [12], и именно с этим связано большинство перечисленных выше проблем.

Также зачастую в вузах отсутствует необходимая инфраструктура для внедрения систем контроля версий, и, если преподаватели хотят их использовать в рамках своих курсов, им необходимо или самостоятельно разворачивать необходимую инфраструктуру на своем оборудовании, или использовать какие-то общедоступные системы.

Для решения части указанных проблем были установлены следующие **требования к внедрению СКВ в процесс обучения:**

- студенты работают в отдельных репозиториях;
- студенты не могут самостоятельно создавать репозитории;
- студенты могут создавать коммиты в свои репозитории;
- студенты не могут вносить изменения в историю в своих репозиториях;
- студенты не имеют доступа к репозиториям друг друга;
- преподаватели могут читать репозитории студентов в рамках своих предметов;
- преподаватели могут создавать репозитории с шаблонами заданий в рамках своих предметов;
- преподаватели могут разом раздать задание путем его копирования каждому студенту;
- преподаватели могут сразу скачать все репозитории студентов.

В результате была разработана система ограничений и разделения прав среди преподавателей и студентов, которая позволяет максимально ограничить им права в выбранной СКВ, оставляя минимальный набор прав, достаточный для работы с нужными репозиториями.

4. Система управления репозиториями GitLab

Перед тем как начать рассматривать предлагаемую систему ограничений, необходимо разобраться с основными понятиями, особенностями и ограничениями, которые присущи самим системам управления СКВ. Системой, на примере которой будет проведен данный анализ и которая была выбрана в качестве системы хостинга Git-репозиториев, выступает GitLab. Она является одной из наиболее популярных систем управления репозиториями Git. GitLab имеет бесплатную версию, доступную любым желающим без ограничений по количеству пользователей, которую можно развернуть на серверах организации [16, 17].

Git-репозитории в GitLab создаются в рамках **проектов**, причем один проект может иметь только один репозиторий.

Проекты могут иметь один из следующих уровней видимости:

- *публичный* (Public) — доступен всем и не требует аутентификации;
- *внутренний* (Internal) — доступен только пользователям сервера GitLab;
- *приватный* (Private) — доступен только владельцу и тем пользователям, кому явно предоставлен доступ.

* Коммит — сохраненный в репозитории набор изменений в программном коде.

Проекты могут быть созданы как напрямую у одного из пользователей в его личном пространстве имен, так и в рамках пространства имен какой-либо группы. Группы позволяют объединить связанные проекты и для этих проектов устанавливать права на уровне группы. Помимо этого можно создать подгруппы в рамках какой-либо группы и выстраивать иерархию из групп (GitLab поддерживает до 20 уровней вложенности).

В рамках конкретного проекта или группы пользователю может быть предоставлен один из следующих уровней прав:

- *Guest* — нет доступа к репозиторию, ограниченный доступ к задачам (issues) в проекте;
- *Reporter* — доступ на чтение к репозиторию, полноценный доступ к задачам в проекте;
- *Developer* — ограниченный доступ на запись к репозиторию;
- *Maintainer* — полноценный доступ на запись к репозиторию, ограниченный доступ к управлению проектом. Самый высокий уровень доступа, который может быть установлен на уровне проекта;
- *Owner* — полноценный доступ к управлению проектом, может быть явно установлен только для пользователей групп, а для проектов данный уровень доступа может быть только унаследован.

Проект наследует пользователей группы, в которой он был создан с их правами в ней. Если проект был создан как личный, то он наследует одного пользователя — создателя проекта с уровнем доступа Owner. При этом можно дополнительно добавить других пользователей либо повысить уровень доступа для унаследованных. Понизить же унаследованный уровень доступа или лишить пользователя унаследованного доступа целиком невозможно.

На диаграмме, представленной на рисунке 1, показаны основные сущности GitLab и связи между ними.

Учитывая особенности и структуру GitLab и установленные требования к внедрению СКВ, **работа со студентами может быть организована следующим образом:**

1. Студентам и преподавателям отключается возможность создания личных проектов. Поскольку в личных проектах их создателю всегда наследуется уровень доступа Owner и он не может быть ограничен, какие-либо ограничения прав для таких проектов будут иначе невозможны. Достигается путем установки максимального количества созданных проектов равным нулю в настройках пользователя.
2. Студентам и преподавателям отключается возможность создания групп, поскольку в противном случае ограничение на создание проектов

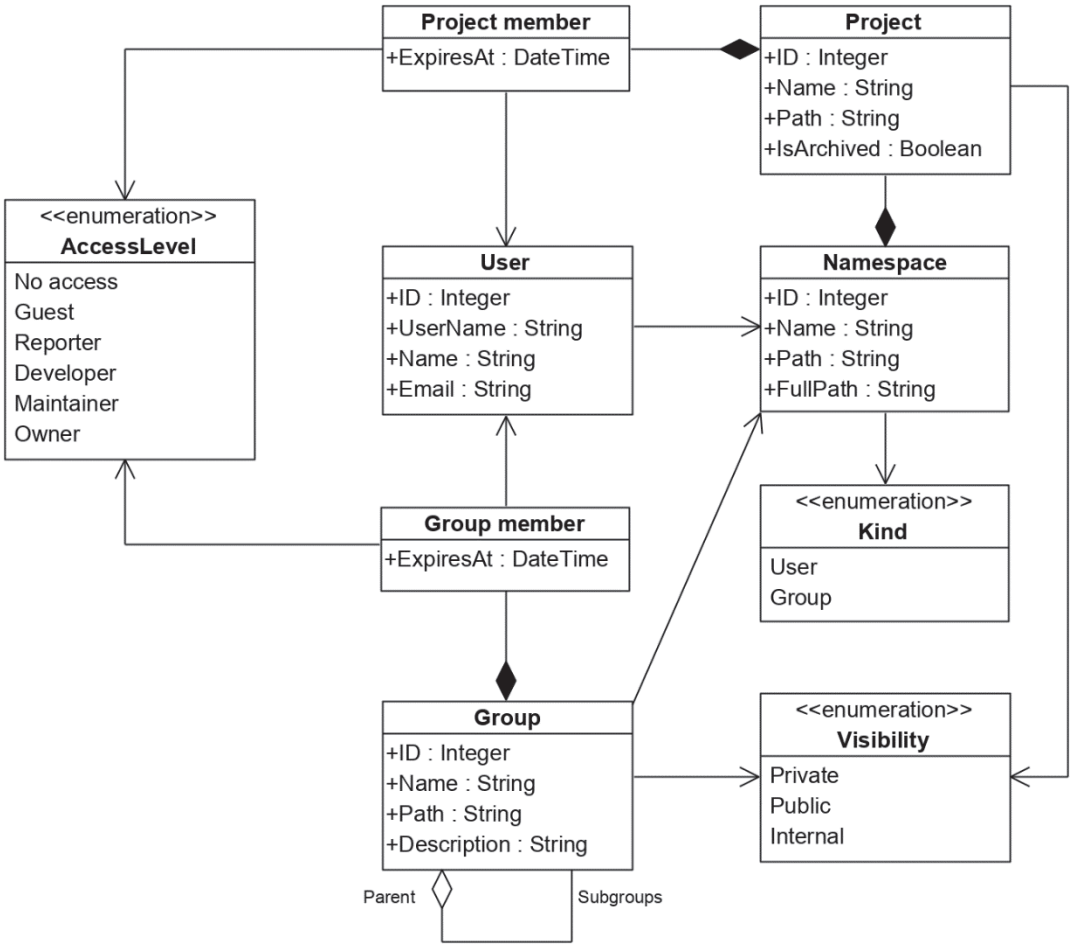

Рис. 1. Основные сущности GitLab

Таблица 2

Особенности различных видов групп: роли пользователей в них и поддерживаемые типы проектов

Особенности	Личная группа студента	Предметная группа приватная	Предметная группа публичная	Предметная группа командная
Участники группы	Студент (D)	Преподаватели (D)	Преподаватели (D) и студенты (R)	Преподаватели (R)
Участники проектов	Студент (D) Преподаватели предмета (R)	Преподаватели (D)	Преподаватели (D) и студенты (R)	Преподаватели (R) Студенты команды (D)
Типы проектов	Личные проекты	Базовый проект Проект с решением Проект с вариантом	Публичные материалы	Командные проекты

можно будет обойти через группы. Достигается путем установки соответствующей опции в настройках пользователя, а также путем установки значения по умолчанию в настройках GitLab.

3. Для каждого студента создается приватная группа, в которой будут создаваться все его личные проекты по всем предметам (подробнее о группах см. табл. 2). Студент добавляется в эту группу с правами Developer, что позволит ему загружать в репозитории свой код, но не позволит выполнять с этим кодом деструктивные действия (выполнять `force push`).
4. Для приватной группы студента изменяется необходимый уровень доступа для создания проектов на Maintainer (более высокий уровень доступа, чем Developer, который назначается студентам), что не позволит студентам самостоятельно создавать проекты в своих группах.
5. В проекты студентов преподаватели добавляются с правами Reporter, что позволяет им просматривать код студентов, но не дает возможности вносить какие-либо изменения.
6. Для каждого предмета создается приватная группа для преподавателей, в которой будут создаваться проекты для заданий, причем на одно задание может быть создано несколько проектов:
   - *базовый проект* — шаблон, на основании которого будут созданы студенческие репозитории;
   - *проект с решением* — созданный на основе шаблона проект, в котором преподаватель может разместить код полного решения задания;
   - *проект с вариантом для контрольной* — созданный на основе шаблона проект, в котором преподаватель сможет разместить дополнительные к шаблону заготовки, специфичные для того или иного варианта контрольной работы.
7. Для каждого предмета создается приватная группа для преподавателей и студентов, в которой преподавателями могут быть размещены какие-либо материалы, которые должны быть доступны для студентов.
8. При необходимости командной работы для каждого предмета также создается приватная группа для командных проектов. Преподаватели добавляются в саму группу, а студенты добавляются явно в проекты их команд.

Данный подход оставляет и студентам, и преподавателям только минимальный набор возможностей в системе и минимизирует возможные деструктивные действия, которые могут привести к повреждению или потере данных. Например, **студентом могут быть совершены следующие действия:**

- потеря доступа к учетной записи;
- выход из личной группы или из публичной предметной группы;
- выход из командного проекта.

Все эти действия могут быть обращены администратором и не ведут к потере данных.

При этом появляется другая проблема: администратору добавляются задачи по сопровождению предметов, поскольку только у него остается возможность создавать проекты. В результате по запросу преподавателя администратору нужно будет создавать проекты, копировать их студентам и выполнять другие действия по инициативе преподавателя.

Для решения данной проблемы был создан **дополнительный инструмент EduRepos** [18], предназначенный для делегации преподавателям необходимых им прав. Этот инструмент предоставляет преподавателям возможность выполнить необходимые им действия, требующие прав администратора в рамках тех предметов, к которым преподаватели имеют доступ. Также данный инструмент автоматизирует массовые действия, например, копирование студентам репозиториев, сбор готового кода из студенческих репозиториев и другие действия, которые даже при наличии необходимых прав в самом GitLab могли бы требовать существенного времени для выполнения при большом количестве студентов.

5. Работа с GitLab и дополнительным инструментом EduRepos

При анализе требований были выделены три вида акторов: студент, преподаватель и администратор. У каждого из них свои задачи, поэтому для каждого актора были заданы функциональные требования,

Таблица 3

Варианты использования (use case)

Актор	GitLab	Дополнительный инструмент EduRepos
Студент	• Просматривать материалы по предмету. • Выполнять домашние и контрольные работы. • Выполнять командные задания. • Добавлять свои материалы и заметки	
Преподаватель	• Публиковать материалы по предмету. • Выкладывать домашние и контрольные работы студентам и командам. • Проверять домашние и контрольные работы студентов и команд	• Управлять проектами по предмету (CRUD). • Записывать студентов на предмет. • Добавлять других преподавателей в предмет пли проект и управлять их правами. • Управлять командами проекта. • Управлять репозиториями проекта. • Смотреть статистику работы студентов и команд. • Архивировать проекты, информацию о студентах и командах. • Собирать код проектов
Администратор	• Выполнять настройку GitLab	• Управлять проектами (CRUD). • Управлять предметами (CRUD). • Управлять информацией о преподавателях (CRUD). • Управлять информацией о студентах (CRUD), в том числе импортировать информацию о студентах. • Управлять группами (CRUD). • Управлять факультетами (CRUD). • Управлять синхронизацией с GitLab. • Управлять пользователями

определенные в виде вариантов использования (use case). В таблице 3 представлено, какие действия может совершать каждый актор и с помощью какой системы.

Как видно из таблицы 3, *студент* не взаимодействует с новым инструментом, все необходимые ему действия он делает с помощью GitLab: получает задание от преподавателя, выполняет его и отправляет выполненное задание. Студент также может просматривать материалы дисциплины, которые публикует преподаватель, и работать над командными проектами вместе с другими студентами.

Преподаватель при таком подходе работает с двумя системами. Через GitLab он выполняет все действия, связанные с публикацией заданий, исходного кода, шаблонов заданий или материалов по предмету, а также с проверкой студенческих работ (возможность GitLab создавать pull-request*, оставлять комментарии и обмениваться сообщениями между преподавателем и студентом). Задачи по поддержке предметов, такие как добавление студентов на свой курс, создание и управление проектами, просмотр статистики работ учащихся и другие, преподаватель выполняет в новом инструменте.

Администратор занимается настройкой и поддержанием системы. При первоначальной настройке он добавляет необходимые факультеты и каждый учебный год, перед началом учебы, добавляет новые группы и заносит в них новых студентов. Также администратор ответственен за добавление списка предметов, занесение преподавателей в систему и связывание их с нужными предметами. Администратор имеет полный доступ ко всем студентам, преподавателям, факультетам, предметам, проектам и командам.

Такая модель работы позволяет поддержать предлагаемый способ организации работы с репозиториями. *Студент* может использовать все возможности Git и работать с GitLab, ничего не зная о новом инструменте. *Преподавателю* новый инструмент автоматизирует те задачи, совершать которые исключительно возможностями GitLab было затруднительно и долго, тем самым адаптируя GitLab под образовательные нужды и делая его еще более удобным для обучения программированию. *Администратору* предоставляется удобный интерфейс для того, чтобы работать с большим количеством предметов и студентов.

6. Модель предметной области

На рисунке 2 представлена модель предметной области дополнительного инструмента EduRepos, где изображены основные сущности, их структура, а также связи между ними.

Как видно из диаграммы, права преподавателю можно задавать на двух уровнях: на уровне предмета и на уровне проекта. Такая настройка прав позволяет более гибко разделять преподавателей, например, задавать преподавателя с полным набором прав или ассистента, который может добавляться только с правом чтения в конкретные проекты.

Рассмотрим внимательно организацию работы преподавателя с новым инструментом. Прежде всего

* Pull-request — запрос на применение своих изменений в коде к другой ветке или к другому репозиторию.

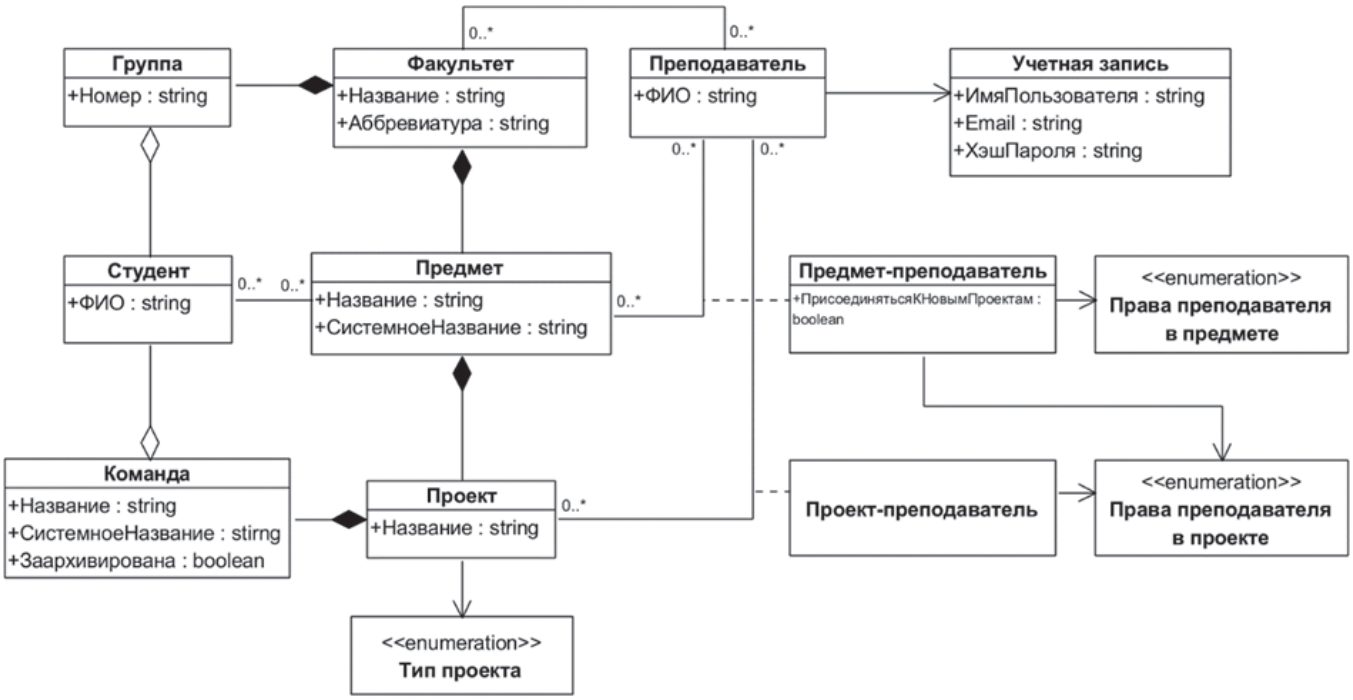

Рис. 2. Модель предметной области инструмента

преподаватель должен добавить нужных студентов на свой курс. После этого он может проводить свой курс — выкладывать нужные материалы в приватную или публичную группу предмета. Преподаватель может создавать новые проекты, которые логически можно понимать как некоторое задание или лабораторную работу, которую студенты должны выполнить, при этом у проекта можно задать срок сдачи. Проекты бывают следующих типов: домашняя работа, контрольная работа, материалы преподавателя, материалы студента, командная домашняя работа, командная контрольная работа, материалы команды.

7. Типы проектов

Разберем подробнее основные типы проектов.

«Домашняя работа» — тип проекта, который позволяет преподавателю подготовить задание для студентов и начальный код, разослать задание и код студентам путем копирования репозитория в их личные группы и затем собрать выполненные студентами задания для проверки (рис. 3).

Создание проекта вида «Домашняя работа» состоит из следующих этапов:

1) создается проект-шаблон в приватной группе (Base Repository);
2) на основании этого шаблона опционально создается репозиторий под решение преподавателя в приватной группе (Completed Repository) путем выполнения копирования репозитория (`fork`);
3) на основании шаблона также создаются репозитории в студенческих группах (Student Repository) путем выполнения копирования репозитория (`fork`).

«Контрольная работа» — тип проекта, позволяющий преподавателю подготовить различные варианты заданий для студентов, которые будут отправлены каждому студенту индивидуально перед началом контрольной (рис. 4).

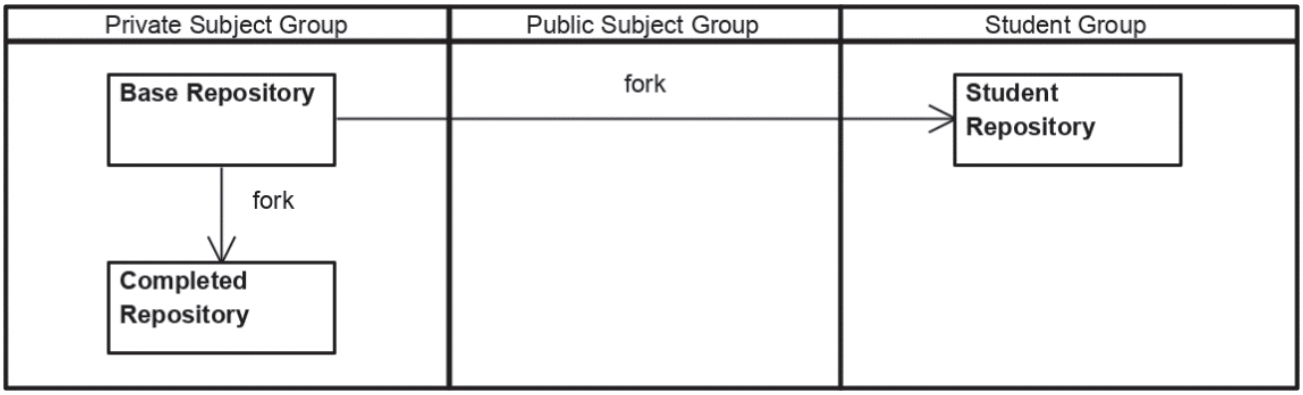

Рис. 3. Проект типа «Домашняя работа»

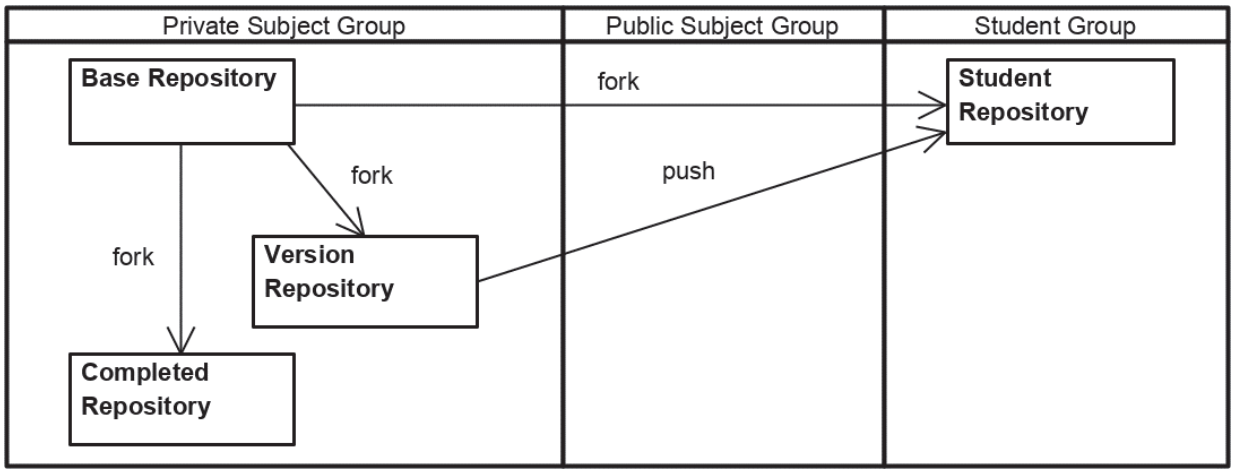

Рис. 4. Проект типа «Контрольная работа»

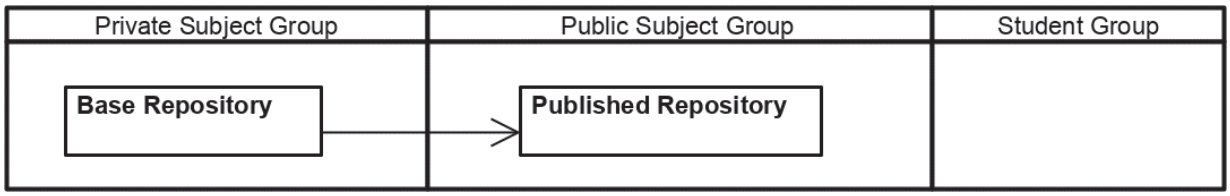

Рис. 5. Проект типа «Материалы преподавателя»

Создание проекта вида «Контрольная работа» состоит из следующих этапов:

1) создается проект-шаблон в приватной группе (Base Repository);
2) на основании этого шаблона опционально создается репозиторий под решение преподавателя в приватной группе (Completed Repository) путем выполнения копирования репозитория (`fork`);
3) на основании шаблона создаются репозитории для каждого варианта в приватной группе (Version Repository) путем выполнения копирования репозитория (`fork`);
4) на основании шаблона также создаются репозитории в студенческих группах (Student Repository) путем выполнения копирования репозитория (`fork`), куда затем, перед началом контрольной работы, с помощью команды `push` студенту отправляется его вариант.

«Материалы преподавателя» — тип проекта, позволяющий преподавателю публиковать материалы для студентов (рис. 5). Для проекта вида «Материалы преподавателя» создается по одному репозиторию в приватной и публичной группах. В приватном репозитории преподаватель готовит материалы, которые затем выкладывает в публичный репозиторий.

Создание проекта вида «Материалы преподавателя» состоит из следующих этапов:

1) создается репозиторий в приватной группе, где преподаватель готовит материалы для студентов (Base Repository);
2) на основании этого репозитория создается репозиторий в публичной группе (Published Repository) путем выполнения копирования репозитория (`fork`).

8. Интеграция с GitLab

Для управления GitLab дополнительный инструмент EduRepos использует REST API, предоставляемый GitLab для взаимодействия с ним [19]. Для этого при первоначальной настройке инструмента ему необходимо указать адрес GitLab, а также Private Access Token администратора, с помощью которого инструмент будет выполнять аутентификацию в API. Также для упрощения использования API была разработана клиентская библиотека, предоставляющая доступ к подмножеству необходимых API-методов, позволяющих манипулировать проектами, пользователями, группами и т. д.

Помимо REST API для отдельных задач также используется прямое взаимодействие с Git-репозиториями [20], размещенными в GitLab, например, для того, чтобы дослать студентам коммиты со специфичной для варианта информацией из репозитория с вариантом. Этот репозиторий сначала клонируется инструментом во временную директорию на сервере, а затем выполняется `push` коммитов в репозиторий студента.

9. Дополнительные возможности

В разработанном инструменте реализован механизм архивации репозиториев, команд и студентов, который позволяет скрыть их без фактического удаления. Такое решение позволяет не загромождать преподавателю интерфейс неактуальными студентами (теми, которые завершили прохождение предмета) и в то же время сохранять все данные. Можно либо необратимо заархивировать репозито-

рии, либо поставить временную блокировку и потом снять ее (например, для того, чтобы студенты не могли вносить изменения в контрольную работу после ее окончания). Студент при архивации сохраняет доступ ко всем своим репозиториям, но теряет возможность вносить изменения в них. При необходимости, если студенту потребуется пройти курс повторно, студент может быть разархивирован преподавателем.

Также новый инструмент позволяет осуществлять сбор выполненных студентами (или командами) заданий — скачать код нескольких репозиториев сразу как один zip-архив.

Помимо этого реализован механизм импорта студентов, который позволяет администратору сразу заносить целые группы студентов в систему, автоматически создавая им пользователей и личные группы.

Для того чтобы отслеживать прогресс студентов, существует возможность просматривать статистику их работы. Статистические данные собираются по каждому предмету в разрезе студентов и проектов. Исходными данными для составления статистики служат коммиты, которые делают студенты. Система периодически запрашивает у GitLab информацию о новых коммитах и кэширует в своей базе данных общую информацию о них, в том числе примерный объем изменений, совершенных в кэшируемом коммите. Преподаватель может посмотреть статистику как по конкретному студенту (общий график активности, последние коммиты), так и по группе студентов в целом. В рамках своего предмета преподаватель может создавать отчеты по статистике работы студентов, представляющие собой таблицы, в которых столбцами являются проекты, строками — студенты, а на их пересечении отображается результат работы студента: количество коммитов в репозитории проектов или суммарный размер изменений в них.

10. Заключение

Внедрение рассмотренного подхода позволяет снизить трудозатраты преподавателя на работу с кодом студентов путем предоставления удобных инструментов для раздачи студентам стартового кода, сбора кода сразу со всех учащихся, просмотра отдельных репозиториев и статистики работы студентов. Студенты получают централизованное место для хранения своего исходного кода с возможностью его синхронизации. При этом предлагаемая модель ограничений позволяет минимизировать возможность повреждения данных, хранящихся в системе, или несанкционированного доступа к ним. Для поддержки предлагаемой модели ограничений был разработан дополнительный инструмент EduRepos, который предполагается использовать вместе с GitLab.

В дальнейшем планируется усовершенствование данного инструмента: реализация более гибкой модели типов проектов, поддержка других систем управления Git-репозиториями, более гибкие возможности делегации прав в системе.

Список использованных источников

1. *Алексеевский П. И.* Применение средств управления версиями для коллективной работы студентов над проектом компьютерной игры // Педагогическое образование в России. 2012. № 6. С. 51–54. http://journals.uspu.ru/attachments/article/307/Педагогическое% 20образование_6_2012_ст.% 2009.pdf
2. git. https://git-scm.com
3. *Chacon S., Straub B.* Pro Git. NYC: Apress, 2014. 440 p. https://git-scm.com/book/en/v2
4. GitHub. https://github.com
5. GitLab. https://about.gitlab.com
6. Bitbucket. https://bitbucket.org
7. Omnibus GitLab Docs. https://docs.gitlab.com/omnibus/
8. *Kelleher J.* Employing git in the classroom // 2014 World Congress on Computer Applications and Information Systems (WCCAIS). IEEE, 2014. P. 1–4. DOI: 10.1109/WCCAIS.2014.6916568
9. *Feliciano J., Storey M.-A., Zagalsky A.* Student experiences using GitHub in software engineering courses: a case study // ICSE '16: Proc. 38th Int. Conf. on Software Engineering Companion. IEEE, 2016. P. 422–431. DOI: 10.1145/2889160.2889195
10. *Biňas M.* Version Control System in CS1 course: Practical experience // 2013 IEEE 11th Int. Conf. on Emerging eLearning Technologies and Applications (ICETA). IEEE, 2013. P. 23–28. DOI: 10.1109/ICETA.2013.6674398
11. *Francese R., Gravino C., Risi M., Scanniello G.* On the experience of using Git-Hub in the context of an academic course for the development of apps for smart devices // The 21st Int. Conf. on Distributed Multimedia Systems. 2015. P. 292–299.
12. *Reid K. L., Wilson G. V.* Learning by doing: Introducing version control as a way to manage student assignments // SIGCSE '05: Proc. 36th ACM technical symposium on Computer science education. ACM, 2005. P. 272–276.
13. *Lawrance J., Jung S., Wiseman C.* Git on the cloud in the classroom // SIGCSE '13: Proc. 44th ACM technical symposium on Computer science education. ACM, 2013. P. 639–644. DOI: 10.1145/2445196.2445386
14. *Clifton C., Kaczmarczyk L. C., Mrozek M.* Subverting the fundamentals sequence: using version control to enhance course management // ACM SIGCSE Bulletin. 2007. Vol. 39. Is. 1. P. 86–90. DOI: 10.1145/1227504.1227344
15. *Zagalsky A., Feliciano J., Storey M.-A., Zhao Y., Wang W.* The emergence of GitHub as a collaborative platform for education // CSCW '15: Proc. 18th ACM Conf. on Computer Supported Cooperative Work & Social Computing. ACM, 2015. P. 1906–1917. DOI: 10.1145/2675133.2675284
16. *Valdivia R. G. B.* Collaborative learning using git with GitLab in students of the engineering programming course // Proc. of the Int. Congress on Educational and Technology in Sciences. 2019. P. 92–101. http://ceur-ws.org/Vol-2555/paper8.pdf
17. *Сидякин И. М.* Применение системы контроля версий GitLab для обучения программированию // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. 2016. № 10. С. 168–179. http://technomag.edu.ru/doc/848154.html
18. *Протасевич Ю. А.* Разработка системы для автоматизации управления Git-репозиториями на базе системы GitLab для использования в процессе обучения: выпускная бакалаврская работа по направлению подготовки: 09.03.04 — Программная инженерия. Томск, 2019. http://vital.lib.tsu.ru/vital/access/manager/Repository/vital:9089
19. API Docs // GitLab. https://docs.gitlab.com/ee/api/
20. libgit2sharp. https://github.com/libgit2/libgit2sharp

TOOLS FOR ORGANIZING TEACHERS-STUDENTS INTERACTION USING VERSION CONTROL SYSTEMS

Yu. A. Protasevich[1], O. A. Zmeev[1], D. A. Sokolov[1]

[1]National Research Tomsk State University
634050, Russia, Tomsk, prospekt Lenina, 36

Abstract

The article describes an approach to organizing the teacher-students interaction in programming courses using the Git version control system. In order to select the most suitable and affordable system for educational needs a comparative analysis of different Git repository management systems was carried out. Based on the experience of various educational institutions that use version control systems in their courses, the advantages and disadvantages of using these systems in teaching were identified. Taking into account the existing problems, a software solution was developed based on the GitLab system. As part of this solution, a method is proposed for organizing the work of a teacher and students in disciplines that use version control systems. This approach implies using both GitLab and additional system, which serves as a manager for Git repositories and is designed to facilitate the work of the teacher and administrator by automating the tasks they perform. The main purpose of the article is a detailed description of this approach: limiting permissions to both teachers and students, GitLab organization and functionality, a list of use cases for each user. The article also presents common workflows of the additional system, its main entities and their relationships and an overview of the features that the system provides.

Keywords: learning process automation, version control systems, Git, GitLab.

DOI: 10.32517/0234-0453-2021-36-4-36-46

For citation:

Protasevich Yu. A., Zmeev O. A., Sokolov D. A. Instrumenty dlya organizatsii vzaimodejstviya prepodavatelej i studentov s ispol'zovaniem sistem kontrolya versij [Tools for organizing teachers-students interaction using version control systems]. *Informatika i obrazovanie — Informatics and Education*, 2021, no. 4, p. 36–46. (In Russian.)

Received: January 31, 2021.

Accepted: April 6, 2021.

About the authors

Yuliya A. Protasevich, a master student at the Software Engineering Department, Institute of Applied Mathematics and Computer Science, National Research Tomsk State University, Tomsk, Russia; protasevich.yuliya@gmail.com; ORCID: 0000-0002-0054-9138

Oleg A. Zmeev, Doctor of Sciences (Physics and Mathematics), Professor, Professor at the Software Engineering Department, Institute of Applied Mathematics and Computer Science, National Research Tomsk State University, Tomsk, Russia; ozmeyev@gmail.com; ORCID: 0000-0002-8170-4290

Danila A. Sokolov, Head of the Digital Solutions Department, National Research Tomsk State University, Tomsk, Russia; danila.sokolov@accounts.tsu.ru; ORCID: 0000-0001-7992-1205

References

1. *Alekseevsky P. I.* Primenenie sredstv upravleniya versiyami dlya kollektivnoj raboty studentov nad proektom komp'yuternoj igry [Using version control systems for students' collaborative work on the computer game development]. *Pedagogicheskoe obrazovanie v Rossii — Pedagogical Education in Russia*, 2012, no. 6, p. 51–54. (In Russian.) Available at: http://journals.uspu.ru/attachments/article/307/Педагогическое% 20образование_6_2012_ст.% 2009.pdf

2. git. Available at: https://git-scm.com

3. *Chacon S., Straub B.* Pro Git. NYC, Apress, 2014. 440 p. Available at: https://git-scm.com/book/en/v2

4. GitHub. Available at: https://github.com

5. GitLab. Available at: https://about.gitlab.com

6. Bitbucket. Available at: https://bitbucket.org

7. Omnibus GitLab Docs. Available at: https://docs.gitlab.com/omnibus/

8. *Kelleher J.* Employing git in the classroom. *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. IEEE, 2014, p. 1–4. DOI: 10.1109/WCCAIS.2014.6916568

9. *Feliciano J., Storey M.-A., Zagalsky A.* Student experiences using GitHub in software engineering courses: a case study. *ICSE '16: Proc. 38th Int. Conf. on Software Engineering Companion*. IEEE, 2016, p. 422–431. DOI: 10.1145/2889160.2889195

10. *Biňas M.* Version Control System in CS1 course: Practical experience. *2013 IEEE 11th Int. Conf. on Emerging eLearning Technologies and Applications (ICETA)*. IEEE, 2013, p. 23–28. DOI: 10.1109/ICETA.2013.6674398

11. *Francese R., Gravino C., Risi M., Scanniello G.* On the experience of using Git-Hub in the context of an academic course for the development of apps for smart devices. *The 21st Int. Conf. on Distributed Multimedia Systems*. 2015, p. 292–299.

12. *Reid K. L., Wilson G. V.* Learning by doing: Introducing version control as a way to manage student assignments. *SIGCSE '05: Proc. 36th ACM technical symposium on Computer science education*. ACM, 2005, p. 272–276.

13. *Lawrance J., Jung S., Wiseman C.* Git on the cloud in the classroom. *SIGCSE '13: Proc. 44th ACM technical symposium on Computer science education*. ACM, 2013, p. 639–644. DOI: 10.1145/2445196.2445386

14. *Clifton C., Kaczmarczyk L. C., Mrozek M.* Subverting the fundamentals sequence: using version control to enhance course management. *ACM SIGCSE Bulletin*, 2007, vol. 39, is. 1, p. 86–90. DOI: 10.1145/1227504.1227344

15. *Zagalsky A., Feliciano J., Storey M.-A., Zhao Y., Wang W.* The emergence of GitHub as a collaborative platform for education. *CSCW '15: Proc. 18th ACM Conf. on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, p. 1906–1917. DOI: 10.1145/2675133.2675284

16. *Valdivia R. G. B.* Collaborative learning using git with GitLab in students of the engineering programming course. *Proc. of the Int. Congress on Educational and Technology in Sciences*. 2019, p. 92–101. Available at: http://ceur-ws.org/Vol-2555/paper8.pdf

17. *Sidyakin I. M.* Primenenie sistemy kontrolya versij GitLab dlya obucheniya programmirovaniyu [Using the GitLab version control system to teach programming]. *Nauka i obrazovanie: nauchnoe izdanie MGTU im. N.EH. Baumana — Science & Education: Scientific Edition of Bauman MSTU*, 2016, no. 10, p. 168–179. (In Russian.) Available at: http://technomag.edu.ru/doc/848154.html

18. *Protasevich Yu. A.* Razrabotka sistemy dlya avtomatizatsii upravleniya Git-repozitoriyami na baze sistemy GitLab dlya ispol'zovaniya v protsesse obucheniya: vypusknaya baka-

lavrskaya rabota po napravleniyu podgotovki: 09.03.04 — Programmnaya inzheneriya [Development of a system for automating the management of Git repositories based on the GitLab system for use in the learning process: graduate bachelor's work in the field of training: 03/09/04 — Software engineering]. Tomsk, 2019. (In Russian.) Available at: http://vital.lib.tsu.ru/vital/access/manager/Repository/vital:9089

19. API Docs *GitLab*. Available at: https://docs.gitlab.com/ee/api/

20. libgit2sharp. Available at: https://github.com/libgit2/libgit2sharp

НОВОСТИ

Microsoft создала инструмент для написания ПО вообще без навыков программирования

В платформу Microsoft Power Apps будет встроена поддержка инструментов с искусственным интеллектом, которые позволят создавать бизнес-приложения без глубоких навыков программирования, используя лишь диалоговые команды.

Искусственный интеллект поможет разработчикам.

Microsoft интегрирует технологии искусственного интеллекта со своим языком программирования Power Fx, который применяется в разработке приложений на платформе Power Platform. Это позволит клиентам компании создавать программы практически без необходимости написания кода. Об этом компания сообщила в рамках своей технологической конференции Microsoft Build 2021. Новые функции будут доступны в рамках публичного предварительного тестирования к концу июня 2021 года на территории Северной Америки. В будущем Microsoft также планирует интегрировать язык Power Fx в другие инструменты Power Platform.

Программирование на естественном языке.

Microsoft объявила о внедрении модели естественного языка OpenAI GPT-3 в платформу для разработки приложений с минимумом программирования Power Apps. Благодаря интеграции пользователи платформы Power Apps смогут создавать приложения в формате диалога с компьютером, поясняют в Microsoft. Например, при разработке приложения в сфере электронной коммерции можно будет описать в диалоге желаемую цель на естественном английском языке: «find products where the name starts with 'kids'» («найти продукты, название которых начинается со слова kids»).

Модель GPT-3 предложит варианты преобразования запроса в формулу Microsoft Power Fx, языка программирования Power Platform. Пользователю же останется только выбрать наиболее подходящий вариант, например «Filter('BC Orders' Left('Product Name', 4)="Kids")».

Несмотря на простоту языка Power Fx формирование, к примеру, сложных запросов к данным все еще может требовать достаточно глубоких технических знаний, по крайней мере, понимания логики написания формул. Использование естественного языка в процессе создания приложений, по мнению специалистов Microsoft, позволит еще больше снизить порог вхождения в разработку приложений.

В Microsoft подчеркивают, что нововведение не заменяет необходимость понимания человеком кода, который он внедряет, а нацелено на помощь людям, изучающим язык программирования Power Fx, и упрощение выбора правильных формул для получения нужного результата.

GPT-3 (Generative Pre-trained Transformer) — крупнейшая языковая модель в мире, разработанная OpenAI для решения любых задач на английском языке. OpenAI является некоммерческой исследовательской организацией, основателями которой выступают главный исполнительный директор Tesla Илон Маск (Elon Musk) и Сэм Альтман (Sam Altman). GPT-3 работает в фирменном облаке Microsoft Azure, а для ее дообучения под задачу был использован сервис Azure Machine Learning.

Интеграция PROSE.

Помимо интеграции с GPT-3 Microsoft планирует дать возможность разработчикам бизнес-приложений Power Apps писать код в рамках концепции «программирование на основе примера» (Programming by Example — PBE). В ней искусственный интеллект генерирует программный код для преобразования данных на базе шаблона, который строит, предварительно проанализировав пользовательский пример — исходную информацию и конечный результат.

В качестве иллюстрации принципа работы техники PBE Microsoft приводит ситуацию, которая могла бы возникнуть в процессе создания приложения для электронной коммерции. Допустим, разработчику необходимо поменять формат отображения имен клиентов в некоторой таблице данных — вместо имени и фамилии теперь должны отображаться имя и инициал, заканчивающийся точкой. Чтобы это реализовать на практике, разработчику достаточно «скормить» системе исходное и желаемое значения, например, «John Snow» и «John S.», после чего она сгенерирует формулу на языке Power Fx (весьма громоздкую в данном случае) на основе выявленных искусственным интеллектом закономерностей. Эта формула позволит преобразовать все оставшиеся данные по заданному шаблону.

За реализацию принципа PBE отвечает технология PROSE (Program Synthesis Using Examples), разработанная командой исследовательского подразделения Microsoft Research.

Power Fx и Power Automate Desktop.

Power Fx — это язык программирования, предназначенный для настройки процессов в Power Platform. Язык основан на синтаксисе функций табличного редактора Microsoft Excel и относится к категории так называемых low-code-инструментов, т. е. не требующих от пользователя серьезных навыков программирования для успешного применения. Код интерпретатора Power Fx открыт и опубликован на хостинге ИТ-проектов Github.

Microsoft впервые объявила о запуске Power Fx в марте 2021 года. Ожидается, что Power Fx поможет снизить порог вхождения в разработку и позволит бизнес-пользователям создавать приложения самостоятельно. Профессиональные же разработчики смогут с его помощью ускорить процесс разработки.

(По материалам CNews)