УДК 004.423

Коломоєць Г.П., к.ф.-м.н., доцент
Таврійський державний агротехнологічний університет
імені Дмитра Моторного,

ОРГАНІЗАЦІЯ КОМАНДНОЇ РОБОТИ СТУДЕНТІВ-ПРОГРАМІСТІВ ПРИ ВИВЧЕННІ ДИСЦИПЛІНИ «ГРУПОВА ДИНАМІКА ТА КОМУНІКАЦІЇ»

Анотація. *У статті розглядається підхід до організації командної роботи студентів-програмістів першого курсу в межах дисципліни «Групова динаміка та комунікації». Запропоновано методику поєднання формування навичок професійної комунікації з опануванням базових практик колективної розробки програмного забезпечення із використанням платформи GitHub та системи контролю версій Git. Описано етапи формування команд, дослідження предметної області, проєктування архітектури програмного забезпечення, організації спільної роботи з кодом та документацією. Особливу увагу приділено використанню GitHub Wiki, Markdown, JavaDoc, роботі з Git. Запропонований підхід дозволяє формувати у студентів молодших курсів навички командної взаємодії та базові компетентності сучасної інженерії програмного забезпечення.*

Ключові слова: *командна розробка програмного забезпечення, GitHub, Git, професійна комунікація, командна робота, GitHub Wiki, JavaDoc, Markdown.*

Постановка проблеми. Сучасна індустрія розробки програмного забезпечення характеризується високим рівнем колективної взаємодії учасників проєкту, що зумовлено складністю програмних систем, необхідністю розподілу функціональних завдань між розробниками та потребою забезпечення безперервного супроводу програмних продуктів на всіх етапах життєвого циклу. У професійній діяльності програміста дедалі більшого значення набувають не лише технічні компетентності, пов’язані з програмуванням, а й навички командної роботи, професійної комунікації, спільного прийняття рішень, документування результатів діяльності та використання сучасних інструментів колективної розробки [1].

Разом із тим організація командної роботи студентів-програмістів на початкових етапах навчання супроводжується низкою педагогічних та

методичних труднощів. Дисципліна «Групова динаміка та комунікації» викладається на першому курсі підготовки бакалаврів (у другому семестрі), коли студенти ще не опанували об'єктно-орієнтоване програмування, технології проєктування програмних систем, шаблони архітектури програмного забезпечення та інші профільні дисципліни. На цьому етапі здобувачі освіти володіють лише базовими знаннями мови програмування Java, що обмежує можливість реалізації складних програмних проєктів та використання повного спектра професійних практик командної розробки.

За таких умов виникає проблема пошуку ефективних підходів до організації командної діяльності студентів, які, з одного боку, забезпечували б формування базових навичок професійної комунікації та колективної взаємодії, а з іншого – дозволяли б поступово ознайомлювати студентів із сучасними інструментами підтримки життєвого циклу програмного забезпечення без надмірного ускладнення технічної складової проєктів.

Наукова новизна запропонованого підходу полягає в інтеграції дисципліни комунікативного спрямування з практикою командної розробки програмного забезпечення на основі GitHub уже на першому курсі підготовки бакалаврів-програмістів.

Аналіз останніх досліджень і публікацій. Питання організації командної розробки програмного забезпечення та формування навичок професійної взаємодії майбутніх ІТ-фахівців активно досліджуються як у сфері інженерії програмного забезпечення, так і в педагогіці вищої школи. Сучасні дослідження підкреслюють, що ефективність створення якісних програмних продуктів значною мірою залежить від командної роботи, координації дій учасників проєкту та використання відповідних засобів підтримки колективної діяльності [2].

Окремий напрям досліджень пов'язаний із використанням платформи GitHub та системи контролю версій Git у професійній та освітній діяльності [3]. Зокрема, у своїх дослідженнях M. Beckman та співавтори наголошують, що використання Git і GitHub у навчальному процесі сприяє формуванню навичок відтворюваності результатів, командної взаємодії та організації професійного робочого процесу [4].

Водночас аналіз наукових публікацій свідчить, що більшість існуючих підходів орієнтована на студентів старших курсів, які вже володіють знаннями об'єктно-орієнтованого програмування, методів проєктування програмних систем та професійних технологій розробки [5]. Недостатньо дослідженими залишаються питання організації командної

роботи студентів першого курсу, які мають лише базові знання програмування та ще не готові до використання складних інженерних практик.

Крім того, у наявних дослідженнях недостатньо уваги приділяється інтеграції дисциплін комунікативного спрямування з практикою командної розробки програмного забезпечення. Потребують подальшого опрацювання методичні підходи до використання *GitHub Wiki* для спільного дослідження предметної області та проєктування архітектури програмного продукту, а також організації колективної роботи з кодом і документацією в межах навчальних проєктів студентів молодших курсів.

Формулювання цілей статті. Метою цієї публікації є аналіз досвіду організації командної розробки проєктів програмного забезпечення студентами-програмістами першого курсу бакалаврату з використанням платформи GitHub та технології Git в рамках дисципліни «Групова динаміка та комунікації».

Виклад основного матеріалу досліджень. Організація командної роботи студентів-програмістів у межах дисципліни «Групова динаміка та комунікації» здійснювалася з урахуванням необхідності поєднання формування навичок професійної комунікації, командної взаємодії та базових практик колективної розробки програмного забезпечення. Запропонований підхід орієнтований на студентів першого курсу, які володіють лише базовими знаннями мови програмування Java та ще не вивчали дисципліни, пов’язані з об’єктно-орієнтованим програмуванням, проєктуванням програмних систем і технологіями розробки програмного забезпечення.

На початку вивчення дисципліни здійснювалося формування команд розробників (Рис. 1). Для забезпечення ефективної організації роботи та підтримки внутрішньокомандної взаємодії спочатку визначалися лідери команд. Кандидатури лідерів пропонувалися викладачем дисципліни попереднього семестру «Основи програмування», який мав можливість оцінити рівень підготовки студентів, їхню активність, відповідальність і здатність до організації колективної діяльності.

Рис. 1. Етапи командної роботи над проєктом програмного забезпечення

Кількість лідерів відповідала кількості навчальних проєктів, запланованих у межах дисципліни (4-5 проєктів з 4-5 розробниками на кожному). Кожен проєкт передбачав розробку консольного застосунку, у якому передбачається взаємодія з користувачем через меню та реалізуються різні алгоритми, які вже опановані студентами (такі, як сортування масивів, чисельне інтегрування, чисельне диференціювання тощо). Після визначення лідерів формування команд здійснювалося шляхом почергового вибору учасників команд кожним лідером. Такий підхід дозволив забезпечити більш збалансований розподіл студентів між командами та створити умови для формування внутрішньої відповідальності учасників за результати спільної діяльності.

Після завершення формування команд викладач надавав доступ усім учасникам відповідної команди до попередньо сформованих окремих репозиторіїв на платформі GitHub, що містили заготовки проєктів мовою програмування Java (головний клас з методом *main*). Репозиторій використовувався як централізоване середовище підтримки навчального проєкту, яке забезпечувало спільну роботу з документацією, вихідним кодом і засобами контролю версій.

На сторінці *GitHub Wiki* викладачем розміщувалися загальний опис проєкту, постановка задачі та перелік функціональних вимог із їх розподілом між абстрактними розробниками. Інструмент *GitHub Discussion* пропонувався командам у якості засобу комунікації для обговорення розподілу завдань між розробниками, визначення архітектури та способів реалізації функцій програми.

На початковому етапі роботи кожна команда створювала сторінку *GitHub Wiki* «Опис предметної області». Основною метою цього етапу було формування у студентів навичок аналізу предметної області, структурування інформації та документування результатів дослідження. Передбачалося, що сторінка *GitHub Wiki* «Опис предметної області» має містити загальну частину з описом предметної області та основними поняттями та визначеннями, за яку відповідав лідер команди, та описом алгоритмів, які використовувалися під час реалізації індивідуальних завдань розробників. Такий підхід дозволив забезпечити занурення кожного учасника команди в загальну проблематику проєкту та одночасно сформувати відповідальність за власну частину роботи.

Окрема увага приділялася вивченню студентами формату *Markdown*, який використовувався для створення та структурування сторінок *Wiki* та файлу *Readme.md* у кодовій базі проєкту. Студенти навчалися використовувати заголовки різних рівнів, списки, таблиці, вставку фрагментів програмного коду, гіперпосилань, схем та зображень. Використання *Markdown* дозволяло формувати навички підготовки технічної документації у форматі, наближеному до професійних практик розробки програмного забезпечення.

Після завершення етапу дослідження предметної області студентам пропонувалося розробити архітектуру програмного проєкту шляхом розподілу методів, що реалізують функціонал програми, між класами та організації взаємодії компонентів системи. Враховуючи те, що студенти ще не вивчали дисципліни, пов’язані з проєктуванням програмних систем, викладачем був розроблений та розміщений у доступному для студентів репозиторії приклад навчального проєкту з розробленим та документованим вихідним кодом і реалізацією сторінок *GitHub Wiki* [6].

Під час проєктування архітектури студенти повинні були підготувати детальний опис структури програмного забезпечення. Опис архітектури розміщувався на сторінці *GitHub Wiki* та містив перелік класів програмного проєкту. Для кожного класу студенти повинні були навести назву класу та опис призначення класу. Також виконувався опис членів класу із зазначенням модифікаторів, типів даних, назв, опису призначення для

полів, модифікаторів доступу, типу значення, що повертається, сигнатури із описом призначення, описом параметрів та описом результату, що повертається для методів.

Крім опису окремих частин системи, студенти повинні були навести загальну логіку роботи програми: порядок взаємодії користувача із системою, послідовність виклику основних функціональних модулів, особливості обробки даних та сценарії використання. Додатково в описі архітектури визначалися критерії готовності програмного продукту, зокрема, реалізація всіх функціональних вимог, коректна взаємодія між модулями, відсутність критичних помилок під час виконання, підтримка валідації даних, що вводяться користувачем, наявність документації. Для візуалізації структури та логіки роботи проєкту студенти створювали блок-схеми програмного забезпечення, які відображали основні етапи роботи, структуру переходів між модулями та послідовність обробки даних. Робота над проєктуванням архітектури дозволила сформувати у студентів первинні навички декомпозиції програмних систем та відповідального розподілу функціональних завдань між частинами системи.

Після затвердження архітектури студенти клонували заготовки проєктів із GitHub-репозиторіїв на локальні комп’ютери. На цьому етапі вони ознайомлювалися з базовими командами Git та принципами роботи розподіленої системи контролю версій. Зауважимо, що викладачем були підготовлені методичні вказівки з описом реалізації основних операцій Git у використовуваному студентами середовищі розробки IntelliJ IDEA [7].

У процесі розробки кожен учасник команди реалізовував власні класи та функціональні модулі, після чого виконував коміти змін до локального репозиторію. Особлива увага приділялася формуванню навичок логічного структурування історії змін та використання змістовних повідомлень комітів.

Окрему увагу приділяли роботі з конфліктами злиття (merge conflicts), які виникали під час одночасного редагування спільних фрагментів коду кількома учасниками команди. Найчастіше такі конфлікти виникали у класі *Main*, який забезпечував організацію взаємодії між окремими функціональними модулями програми. Студенти навчалися аналізувати конфліктні зміни, порівнювати різні варіанти реалізації та вручну виконувати інтеграцію коду із збереженням працездатності програми. Такий підхід сприяв формуванню навичок колективної взаємодії, узгодження технічних рішень та відповідального ставлення до спільного програмного коду.

Обов’язковою вимогою до реалізації програмних модулів була організація валідації даних, що вводяться користувачем. Студенти повинні були передбачити перевірку коректності введених значень, обробку помилкових ситуацій та інформування користувача про некоректне введення даних. Це сприяло формуванню у студентів розуміння важливості забезпечення надійності та стійкості програмного забезпечення.

Крім того, усі програмні модулі повинні були супроводжуватися документацією засобами JavaDoc. Документування включало опис призначення класів, методів, параметрів і значень, що повертаються, а також пояснення особливостей роботи алгоритмів, які були попередньо напрацьовані при документуванні архітектури проєкту. Використання JavaDoc дозволило сформувати у студентів базові навички створення технічної документації програмного забезпечення та підтримки зрозумілості програмного коду для інших учасників команди.

Після завершення розробки та апробації програмного забезпечення команди готували презентацію власного проєкту. Захист проєктів проводився у формі командного представлення результатів роботи перед студентами інших команд. Під час захисту оцінювалися рівень командної взаємодії, якість програмної реалізації, повнота документації, обґрунтованість архітектурних рішень, використання засобів GitHub та здатність студентів аргументовано пояснювати прийняті технічні рішення.

Запропонований підхід дозволив інтегрувати формування навичок професійної комунікації та командної взаємодії з набуттям базового досвіду використання сучасних інструментів колективної розробки програмного забезпечення вже на першому курсі підготовки бакалаврів-програмістів.

Висновки. У статті розглянуто підхід до організації командної роботи студентів-програмістів першого курсу в межах дисципліни «Групова динаміка та комунікації» із використанням платформи GitHub та системи контролю версій Git. Запропонована методика дозволяє поєднати формування навичок професійної комунікації, командної взаємодії та базових практик сучасної інженерії програмного забезпечення вже на початковому етапі підготовки бакалаврів.

У процесі виконання навчальних проєктів студенти опановують навички роботи з GitHub Wiki, Markdown, Git, інтеграцією змін та усуненням конфліктів злиття. Додатково формуються компетентності документування програмного забезпечення засобами JavaDoc, проєктування архітектури програмних систем та організації валідації даних.

Запропонований підхід дозволяє адаптувати практики командної розробки програмного забезпечення до рівня підготовки студентів першого курсу, які ще не володіють повним спектром професійних знань. Використання командних проєктів та засобів GitHub сприяє підвищенню мотивації студентів, розвитку відповідальності за результати колективної діяльності та наближенню навчального процесу до сучасних практик IT-індустрії.

Перспективним напрямом подальших досліджень є розроблення критеріїв оцінювання ефективності командної взаємодії студентів у процесі розробки програмного забезпечення, а також аналіз впливу використання GitHub та Git на формування професійних компетентностей майбутніх фахівців у сфері інформаційних технологій.

Література

1. Pressman R.S, Maxim B.R. Software Engineering: A Practitioner's Approach. 9th ed. New York: McGraw-Hill Education, 2020. 672 p.
2. Пальонна Т.А., Коваленко Д., Пепчук С.М. Залежність успіху розробки програмного забезпечення від якості командної роботи. *Збірник наукових праць Черкаського державного технологічного університету.* Серія: Економічні науки. 2018. №48. С. 74–83.
3. The Emergence of GitHub as a Collaborative Platform for Education / A. Zagalsky, J. Feliciano, M.-A. Storey et al. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing Companion*. 2015. P. 1906–1917.
4. Implementing version control with Git and GitHub as a learning objective in statistics and data science courses / M.D. Beckman, M. Çetinkaya-Rundel, N.J. Horton et al. URL: https://arxiv.org/abs/2001.01988.
5. Haaranen L, Lehtinen T. Teaching Git on the Side: Version Control System as a Course Platform. *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education*. 2015. P. 87–92.
6. ArrayAnalyzer. Зразок виконання навчального проєкту з дисципліни «Групова динаміка та комунікації». *Репозиторій проєкту на GitHub*. URL: https://github.com/ESEI-ZNU/ArrayAnalyzer.
7. Коломоєць Г.П. Розробка групового проєкту програмного забезпечення на платформі GitHub. Анотація технології навчально-методичного комплексу дисципліни «Групова динаміка та комунікації» освітньої програми підготовки бакалаврів «Програмне забезпечення систем» URL: https://github.com/ESEI-ZNU/ArrayAnalyzer/blob/main/Розробка%20групового%20проєкту%20програмного%20забезпечення%20на%20платформі%20GitHub.pdf.

Kolomoiets H. Students-programmers teamwork organization while studying the «Group Dynamics and Communications» course

Summary. *The article considers an approach to organizing teamwork among first-year students-programmers within the discipline "Group Dynamics and Communication". A methodology is proposed for combining the formation of professional communication skills with mastering basic practices of collective software development using the GitHub platform and the Git version control system. The stages of team formation, research of the subject area, design of software architecture, organization of teamwork with code and documentation are described. Particular attention is paid to the use of GitHub Wiki, Markdown, JavaDoc, and working with Git. The proposed approach allows junior students to develop team interaction skills and basic competencies of modern software engineering.*

Keywords: *team software development, GitHub, Git, professional communication, teamwork, GitHub Wiki, JavaDoc, Markdown.*

УДК 378.016:37.035.6

Кордій О.А., заст. директора з навч. роботи, викладач-методист
ВСП «Бердянський фаховий коледж Таврійського державного агротехнологічного університету імені Дмитра Моторного»

РОЛЬ ГУМАНІТАРНОЇ ОСВІТИ У ФОРМУВАННІ НАЦІОНАЛЬНОЇ ІДЕНТИЧНОСТІ ТА ГРОМАДЯНСЬКОЇ ПОЗИЦІЇ МОЛОДІ В УМОВАХ СУЧАСНИХ ВИКЛИКІВ

Анотація*. У статті проаналізовано проблеми гуманітарної освіти в Україні, зроблено акцент на важливості вивчення гуманітарних наук у навчальних закладах фахової передвищої освіти України, оскільки зростає їхня вагома роль у формуванні громадянської позиції, консолідації українського суспільства, розв'язанні проблеми особистісної та національної самоідентифікації. Зроблено висновок, що сучасне суспільство має суттєво підвищити вимоги до рівня знань молоді у галузі соціально-гуманітарної підготовки.*

Ключові слова: *гуманітарна освіта, гуманітарні науки, історія, культура, філософія, глобалізація.*