УДК 378.1:004:303.732.4(477)

Н.М. Сидорова, асп.

ФОРМУВАННЯ ГОТОВНОСТІ БАКАЛАВРІВ З ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДО ПРОФЕСІЙНОЇ КОМУНІКАЦІЇ

Національний авіаційний університет
E-mail: nika.sidorova@gmail.com

Запропоновано формування професійної комунікації у фахівців з інженерії програмного забезпечення. Розглянуто особливості навчання інженерії програмного забезпечення.

Ключові слова: групова динаміка, інженерія програмного забезпечення, навчання, онтології, професійні комунікації.

Постановка проблеми

Індустрія розробки програмного забезпечення в Україні постійно розвивається. Отже, виникає необхідність у фахівцях. У 2006 р. в Україні було засновано новий напрям навчання з підготовки інженерів програмного забезпечення 6.050103 «Програмна інженерія» відповідно до міжнародного стандарту [1], розроблено відповідні стандарти та розпочато викладання дисциплін [2].

П'ять років існування напряму показали, що потребує вирішення проблема розробки інтегрованого підходу до формування готовності майбутніх бакалаврів до професійної комунікації при реалізації процесів програмного забезпечення.

Аналіз публікацій

Сучасний період в економічно розвинених країнах характеризується переходом від суспільства конкретних знань до інформаційно-компетентнісного суспільства. Каталізатором змін є глобалізаційні процеси побудови єдиного світового та європейського інформаційно-освітнього простору [3].

Розробка програмного забезпечення є командною справою.

Групова динаміка і професійні комунікації мають велике значення для ефективного виконання процесів життєвого циклу програмного забезпечення [4–6].

Інженер повинен:

1) володіти:
   - продуктивними знаннями;
   - інтегрованими вміннями професійної комунікації;
   - особливостями вербальних і формальних засобів комунікації;
2) мати:
   - навички культури спілкування;
   - вміння застосувати їх на практиці.

Це, в свою чергу, вимагає від майбутніх бакалаврів з інженерії програмного забезпечення:

- формування здібностей до алгоритмічного мислення;
- сукупності знань та навичок у сфері інформаційно-комп'ютерних технологій;
- здатності використовувати їх у професійній діяльності;
- мотиваційної потреби у постійному професійному самонавчанні та самовдосконаленні в знаннях різних прикладних доменів, оскільки фахівці мають комунікації з замовниками і користувачами з цих доменів.

Метою роботи є:

- теоретичне обґрунтування та експериментальна перевірка організаційно-педагогічних засобів формування у майбутніх фахівців у галузі інженерії програмного забезпечення готовності до професійної комунікації на основі інтегрованого підходу;
- дослідження організаційно-педагогічних методів, моделей формування готовності майбутніх фахівців з інженерії програмного забезпечення до професійної комунікації.

Професійна підготовка майбутніх фахівців

Готовність майбутніх бакалаврів з інженерії програмного забезпечення має поєднувати загальні, особливі та індивідуальні ознаки.

Загальні ознаки зумовлюють впровадження в навчально-виховний процес дидактичної моделі інтегрованої готовності майбутніх бакалаврів заснованої на застосуванні інноваційних технологій.

Особливі ознаки обґрунтовують використання державних стандартів, відповідних освітньо-кваліфікаційних характеристик та освітньо-професійної програми для скоригованості змістового та процесуального компонентів педагогічного впливу.

Індивідуальні ознаки – це посилання мотивації навчання через використання при формуванні готовності майбутніх бакалаврів проектного інтегративного навчального курсу з відповідними засобами інтерактивних технологій.

Понятійним підґрунтям для реалізації підходу є знання, які можуть бути подані у вигляді онтології [7].

Онтологію можна застосовувати як інструмент при навчанні професійної комунікації, оскільки вона містить знання щодо відповідних сутностей і відношень.

Одним із кроків створення онтології є концептуалізація, яка потребує визначення меж онтології, концептів, опису кожного з них, наприклад, глосарію, специфікування властивостей і відношень.

Дослідження літератури

Систематичний огляд літератури (Systematic Mapping Study) – це один із методів дослідження, що останнім часом набуває все більшого поширення у дослідженні літератури з інженерії програмного забезпечення [8].

Систематичний огляд літератури являє собою структурний тип дослідження публікацій та їх класифікування.

Систематичний огляд літератури традиційно починається з побудови протоколу, що визначає завдання та питання дослідження, виходячи з яких підбираються відповідні методики проведення науково-дослідної роботи. Огляд засновується на певній визначеній стратегії, що спрямована на пошук якомога більшої кількості значущої літератури.

Перші етапи Systematic Mapping Study є дуже схожими з етапами традиційних систематичних оглядів літератури:

– пошук: визначення низки науково-дослідницьких робіт, в яких можуть розкриватися поставлені питання/завдання дослідження;

– включення/виключення науково-дослідницьких робіт із дослідницького процесу,відбір літератури для проведення подальшого дослідження;

– відповідність питанням/завданням дослідження та, якщо необхідно, проведення якісного оцінювання віднайдених науково-дослідницьких робіт.

Основна мета систематичного огляду літератури – надання широкого та повного рівня дослідженності заданої сфери, а також визначення кількості й якості надрукованих науково-дослідницьких робіт із теми.

Вторинною метою систематичного огляду літератури є надання чіткого графічного звіту про проведене дослідження.

У проведеному дослідженні визначено три дослідницькі питання (ДП) для проведення систематичного огляду науково-дослідницьких робіт у сфері освіти інженерії програмного забезпечення:

– ДП1: якими є головні характеристики існуючої системи освіти інженерії програмного забезпечення;

– ДП2: якими є основні дидактичні та педагогічні питання освіти інженерії програмного забезпечення, наскільки дослідженими є питання професійної компетентності, акредитації та ліцензування в інженерії програмного забезпечення;

– ДП3: які можливі вирішення існуючих проблем та шляхи поліпшення системи освіти інженерії програмного забезпечення пропонуються.

Одним із найважливіших етапів у систематичному огляді літератури є визначення якісних критеріїв включення/виключення робіт у дослідження.

Критеріями включення робіт до проведеного дослідження є:

– роботи, надруковані англійською, російською, українською мовами;

– науково-дослідницькі роботи, які містять у назві або тексті терміни «освіта інженерії програмного забезпечення», «професійна компетентність інженера програмного забезпечення»;

– кандидатські та докторські дисертації з навчання інженерії програмного забезпечення;

– навчання з нефіксованим часовим періодом;

– «сіра література», включаючи доповіді, надруковані незалежно від освітніх та професійних організацій.

Критеріями виключення робіт із дослідження є:

– науково-дослідницькі роботи, що повторюються в декількох електронних бібліотеках (до наукового дослідження включається тільки одна робота);

– науково-дослідницькі роботи, що мають однакові результати дослідження (до уваги беруть лише одну найбільш повну роботу;

– науково-дослідницькі роботи та книги, що є недоступними для завантаження;

– наукові роботи, що не містять емпіричних досліджень, або література, що доступна лише у вигляді окремих абзаців (параграфів) чи подана у форматі Power Point презентацій;

– роботи, основною темою дослідження яких не є навчання інженерії програмного забезпечення, або задана тематика лише згадується в декількох реченнях/абзацах.

Для автоматичного пошуку літератури визначено шість електронних баз даних (бібліотек):

– IEEEXplore Digital Library;
– ACM Digital Library;
– Elsevier ScienceDirect;
– EI Compendex;
– Scopus;
– Web of Science.

У результаті пошуку в електронних базах даних науково-дослідницьких робіт із 9711 робіт лише 630 були доступні для завантаження.

Після завантаження лише 113 робіт були включені в подальше дослідження відповідно до якісних критеріїв включення/виключення. Оскільки використовували декілька електронних баз даних, багато робіт мали дублікати.

Остаточно кількість наукових робіт було скорочено до 40.

У процесі неавтоматичного пошуку літератури спочатку з 358 публікацій лише 145 були доступні для завантаження і 126 були включені до дослідження.

За результатами неавтоматичного (електронного) та автоматичного пошуку науково-дослідницьких робіт було знайдено 485 публікацій. Однак після перевірки результатів обох пошуків з 239 робіт, включених в остаточне дослідження, було зменшене до 184 через наявність дублікатів.

На останньому етапі систематичного огляду літератури після якісного оцінювання знайдених публікацій було відібрано 93 науково-дослідницьких робіт, що повністю відповідали поставленим критеріям пошуку.

Підсумки процесу пошуку науково-дослідницьких робіт на кожному етапі дослідження наведено в таблиці.

Етапи відбору наукових досліджень

Пошук	Етап			
	I	II	III	IV
Автоматичний (електронний)	9711	630	113	40
Неавтоматичний	358	145	126	53
Загальна кількість робіт	10 069	485	239	93

У процесі синтезу та аналізу знайдених науково-дослідницьких робіт користувалися методиками порівняння робіт, поєднаних з якісним їх аналізом [8].

Публікації та параграфи робіт, що відповідають дослідницьким завданням, були ретельно досліджені згідно з поставленими питаннями та критеріями (рис. 1).

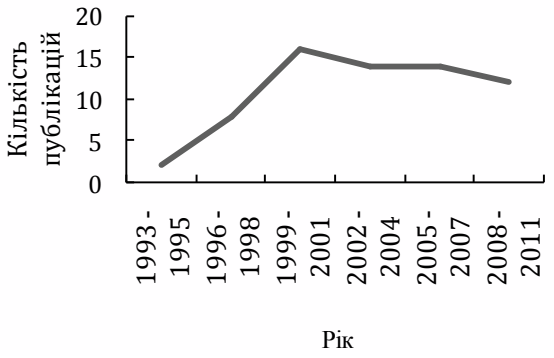

Рис. 1. Класифікація робіт відповідно до року їх публікації

Період 1999–2001 рр. був найпродуктивнішим з погляду дослідження проблем освіти інженерії програмного забезпечення. У першому десятиріччі XXI ст. кількість публікацій знижена, що підтверджує необхідність проведення подальших досліджень.

Найдавніша знайдена публікація з питань навчання інженерії програмного забезпечення датується 1976 р.

Згідно з класифікацією знайдених досліджень з 93 науково-дослідницьких робіт 74 % є емпіричними, тобто заснованими на проведених експериментальних дослідженнях (рис. 2).

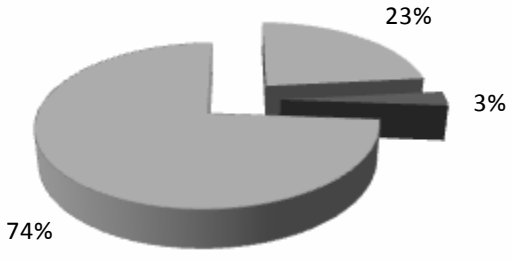

Рис. 2. Типи знайдених науково-дослідницьких робіт

Теоретичні або поняттєві роботи, що базуються на розумінні галузі освіти інженерії програмного забезпечення із власного досвіду вченого (дослідника), становлять 23 %. Незначна кількість робіт (3 %) є оглядами літератури або вторинними дослідженнями, що розглядають уже подані в інших роботах результати.

Класифікацію розглянутих науково-дослідницьких робіт, що присвячені основним параметрам освітньої системи в інженерії програмного забезпечення – загальне значення проведення досліджень у галузі, структура та недоліки існуючої освітньої системи, деталізовано в роботі [9].

Загальна характеристика системи викладання інженерії програмного забезпечення включає такі ДП1:

– значення проведення досліджень у галузі навчання інженерії програмного забезпечення [8; 9];

– структуру існуючої системи викладання інженерії програмного забезпечення [9];

– недоліки існуючої системи викладання інженерії програмного забезпечення [9; 10].

До основних педагогічних питань, що висвітлюються у науково-дослідницьких роботах у сфері викладання інженерії програмного забезпечення, належать ДП2:

– дидактичні та педагогічні питання викладання інженерії програмного забезпечення [9];

– методики та підходи, що використовуються у системі викладання інженерії програмного забезпечення [3; 4; 8; 9; 10];

– проблеми професійної комунікації інженерів програмного забезпечення [9];

– уміння та знання, необхідні для професійної комунікації інженера [9];

– питання ліцензування, сертифікації та акредитації в галузі інженерії програмного забезпечення [9].

Підходи та методики поліпшення якості викладання інженерії програмного забезпечення для підвищення професійної компетентності фахівців галузі включають ДП3:

– навчання, поєднане з практикою на підприємствах [2; 6; 9];

– групові методики навчання [1; 2; 5; 7; 9; 10];

– участь у довільних проектах [2; 9];

– самонавчання [9].

Проведене дослідження є першим у вітчизняному навчанні інженерії програмного забезпечення.

Для майбутнього розвитку освітньої системи необхідно проведення подальших вторинних досліджень та літературних оглядів.

Онтологія

Приймаючи до уваги місце бакалавра в робочому процесі, згідно з інженерією програмного забезпечення загальна онтологію відповідно до стандарту IDEF5 розглядали в чотирьох аспектах (рис. 3).

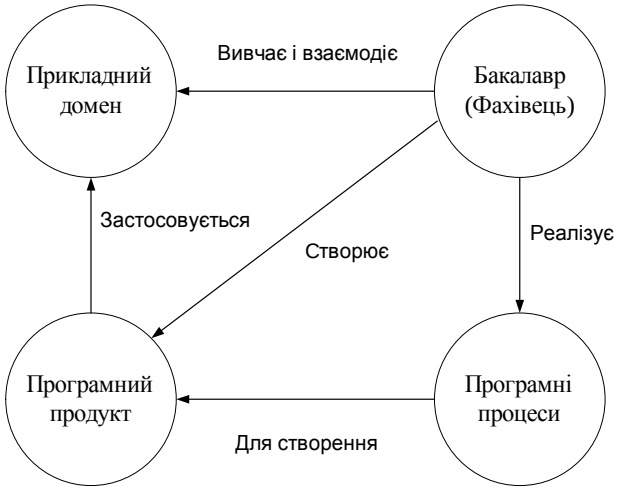

Рис. 3. Загальна онтологія

Фахівець реалізує програмні процеси, які спрямовані на створення програмного продукту, для застосування в прикладному домені. При цьому фахівець взаємодіє з суб'єктами прикладного домену і повинен мати відповідні комунікативні знання та вміння.

Опис загальної онтології містить:

– вербальний опис;

– графічне зображення;

– аналітичний опис.

Вербальний опис містить:

– бакалавр (фахівець) – випускник вищого навчального закладу відповідного напрямку навчання;

– прикладний домен – область реального світу, для якої створюється програмний продукт;

– програмні процеси – процеси, які виконуються протягом життєвого циклу та спрямовані на створення програмного продукту;

– програмний продукт – програмне забезпечення, яке застосовується в прикладному домені і відповідає вимогам**.**

Графічне зображення показано на рис. 3.

Аналітичний опис містить:

– системи та елементи;
– унікальні ідентифікатори;
– з'єднувачі елементів;
– спостереження та зведення;
– коректність;
– аксіоми.

Для аналітичного опису застосовано метод, наведений у роботі [10].

Система та її елементи. Розглядається система s:S з бакалавром (фахівцем) a:A, з прикладним доменом a1:A1, з програмним продуктом a2:A2, з програмними процесами a3:A3, де S, A, A1, A2, A3 – типи.

Елементи u:U, де U – універсальна множина, є або бакалавром (фахівцем), або прикладним доменом, або програмним продуктом, або програмними процесами:

U=A|A1|A2|A3.

Елементи пояснюються в термінах незв'язних типів: бакалавр (фахівець) AA, прикладний домен AO, програмний продукт AB, програмні процеси AC:

A=mkA (aa: AA)
A1=mkA1(ao:AO)
A2=mkA2(ab: AB)
A3=mkA3(ac:AC)

Унікальні ідентифікатори. З кожним елементом пов'язується унікальний ідентифікатор ui:UI, де UI – тип.

З елементу може прослідковуватися його унікальний ідентифікатор:

obsUI:U->UI.

Отже, з елементу може прослідковуватися або бакалавр (фахівець), або прикладний домен, або програмний продукт, або програмні процеси:

is_A:U->Bool
is_A(u)≡case 1 of mkA(_)->true,_->false end
is_A1:U->Bool
is_A1(u)≡case 1 of mkA1(_)->true,_->false end
is_A2:U->Bool
is_A2(u)≡case 1 of mkA2(_)->true,_->false end
is_A3:U->Bool
is_A3(u)≡case 1 of mkA3(_)->true,_->false end

З'єднувачі елементів. З бакалавром (фахівцем) асоціюється максимальна кількість вихідних зв'язків m, більша за одиницю та нуль вхідних зв'язків:

obs_inCs:A->{0:Nat}
obs_outCs:A->Nat

З прикладним доменом асоціюється максимальна кількість вхідних зв'язків c, більша за одиницю та нуль вихідний зв'язків:

obs_inCs:A1->Nat
obs_outCs:A1->{0:Nat}

З програмним продуктом асоціюється максимальна кількість вхідних зв'язків d, більша за одиницю та один вихідний зв'язок:

obs_inCs:A2->Nat
obs_outCs:A2->{1:Nat}

З програмними процесами асоціюється один вхідний зв'язок і один вихідний зв'язок:

obs_inCs:A3->{1:Nat}
obs_outCs:A3->{1:Nat}

Аксіоми:

∀ a:A (obs_outCs(a) ≥2)
∀ a1:A1 (obs_inCs(a1) ≥2)
∀ a2:A2 (obs_inCs(a2) ≥2)

Спостереження та з'єднання. З системи можуть прослідковуватися всі її елементи:

obs_Us:S->U-set

З елемента можна спостерігати пари вхідних та вихідних елементів, що не перетинаються, з якими вони поєднані:

obs_cUIs:U->UI-set×UI-set
wf_Conns:U->Bool
wf_Conns(u)≡ let(iuis,ouis)=obs_cUIs(u)
in iuis∩ ouis={0}∧case u of

а) бакалавр (фахівець) може бути поєднаний з нулем вхідних елементів та всіма вихідними елементами:

mkA (_)->card iuis ∈ {0}∧card ouis ∈ {2… obs_outCs(a)}

б) прикладний домен може бути поєднаний з елементами з бакалавра (фахівця) та програмного продукта (вхідні) та нулем вихідних елементів:

mkA1(_)->card iuis ∈ {2… obs_inCs(a1)}∧card ouis ∈ {0}

в) програмний продукт може бути поєднанний з елементами з бакалавра (фахівця) та програмних процесів (вхідні) та з нулем або одним елементом з прикладного домену (вихідні):

mkA2(_)->card iuis ∈ {2… obs_inCs(a2)}∧card ouis ∈ {0,1}

г) програмні процеси можуть бути поєднанні з нулем або одним вхідним елементом з бакалавра (фахівця) та з нулем або одним елементом з програмного продукта (вихідні):

mkA3(_)->card iuis ∈ {0,1}∧card ouis ∈ {0,1}

end end

Коректність. Елементи ідентифікаторів отримані з obs_cUIs спостереженням повинні бути ідентифіковані, як елементи з S.

Аксіома:

∀ s:S,u:U•u∈
obs_Us(s)⇒let(iuis,ouis)=obs_cUIs(s) in

∀ ui:UI•ui∈ iuis∪ ouis⇒∃ u′:U∈
obs_Us(s)∧ u′≠u∧obs_UI(u′)=ui

end

Графічні зображення двох підонтологій за аспектами прикладного домена й бакалавра показано на рис. 4, 5.

Аналітичний опис підонтологій може бути зроблений так, як наведено.

Рис. 4. Підонтологія прикладного домена

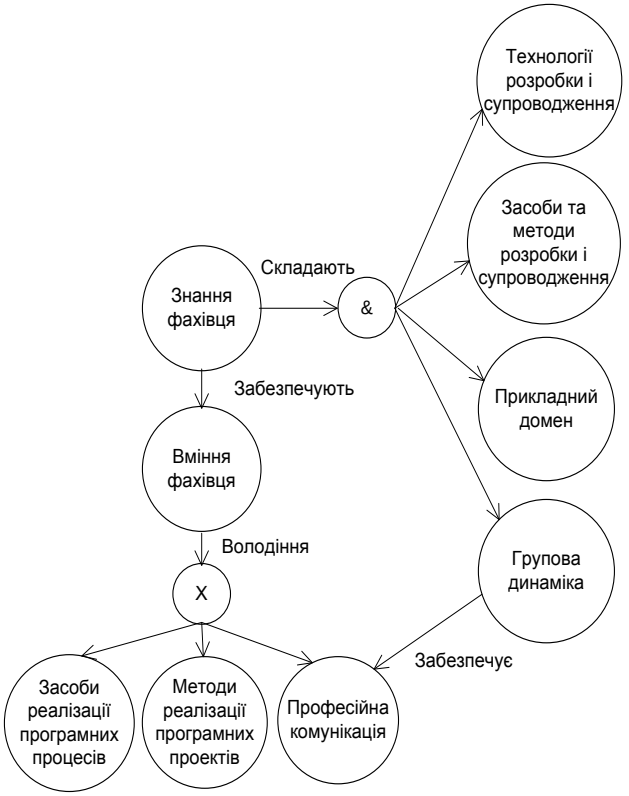

Рис. 5. Підонтологія бакалавра (фахівця)

Підонтологія прикладного домена являє стан знань частини реального світу, де буде використовуватися програмний продукт. При комунікації з фахівцями прикладного домену інженер з розробки програмного забезпечення повинен мати відповідні знання (рис. 4).

Крім знань домену бакалавр повинен мати навички групової динаміки, що забезпечить йому реалізацію відповідної професійної комунікації (рис. 5).

Висновки

За результатами проведеного аналізу навчальної та наукової літератури з урахуванням практичних вимог щодо професійної комунікації майбутніх бакалаврів з інженерії програмного забезпечення побудовано онтологію щодо вирішення завдання формування готовності фахівців до професійної комунікації.

Надалі буде виявлено інваріантний склад технічних і гуманітарних компетенцій, які необхідно сформувати в майбутніх фахівців для забезпечення їх готовності до професійної комунікації в процесі професійної діяльності.

Література

1. *Software* Engineering 2004. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. A volume of the Computing Curricula Series. August 23, 2004. – 128 p.

2. *Модель* випускника бакалаврату "Програмна інженерія" (3 досвіду роботи науково-методичної підкомісії 050103) / М. Бондаренко, М. Сидоров, Т. Морозова, І. Мендзебровський // Вища школа. – 2009. – № 4. – С. 50–61.

3. *Лузік Е.В.* Роздуми про вищу технічну освіту України: наукове ювілейне видання / Е.В. Лузік. – К.: Вид-во «Аспірант», 2011. – 231 с.

4. *Сидоров М.О.* Групова динаміка і комунікації: конспект лекцій / М.О. Сидоров.– К.: Вид-во «НАУ-друк», 2010. – 60 с.

5. *Кириленко Е.Г.* Обоснование содержания обучения в рамках методологии преподавания профессионально-ориентированной дисциплины «Груповая динамика и коммуникация» / Е.Г. Кириленко, О.В. Лучшева // Інженерія програмного забезпечення. – К.: НАУ. – 2010. – № 1.– С. 62–71.

6. *Сидоров М.О.* Професійна практика програмної інженерії – досвід викладання / М.О. Сидоров, І.Б. Мендзебровський, А.А. Орехов // Інженерія програмного забезпечення. – К.: НАУ. – 2010. – № 2. – С. 56–63.

7. *Calero C.* Ontologies for Software engineering and Technology / C. Calero, F. Ruiz, M. Piattini. – Springer. – Berlin. – 2006. – 240 p.

8. *Petersen K.* Systematic Mapping Studies in Software Engineering / K. Petersen, R. Feldt, M. Maltsson // 12th International Conference on Evalution and Assessment in Softare Engibeering, 2008. – P. 378–390.

9. *Сидорова Н.Н.* Навчання інженерії програмного забезпечення – системний огляд літератури / Н.Н. Сидорова // Інженерія програмного забезпечення. – К.: НАУ. – 2011. – № 2 (6). – С. 56–67.

10. *Луцький М.Г.* Онтологія безпеки авіації / М.Г. Луцький, М.О. Сидоров. – К.: НАУ.– 2011. – 412 с.

Стаття надійшла до редакції 19.06.2012.