УДК 502:004.45 (075.8) http://doi.org/10.15407/pp2022.02.022

М.О. Сидоров

ДИСЕРТАЦІЯ МАГІСТРІВ З ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ – ОБЬЄКТ, ПРЕДМЕТ, ЗМІСТ ДОСЛІДЖЕНЬ

На відміну від закордонних університетів, в Україні протягом багатьох років склалася й існує практика формулювання мети дослідження, застосовуючи поняття об'єкта і предмета дослідження. І це, як показав час, важливий етап роботи, від виконання якого залежить результативність дослідження. Загальне й тотальне проникнення програмного забезпечення в життя з одного боку, а з іншого - поява, спеціалізацій інженерії програмного забезпечення значно ускладняють умови виконання цього етапу магістерського дисертаційного дослідження. Крім цього, в інженерії програмного забезпечення застосовуються наукові методи досліджень, які бажано мати на увазі і використовувати для виконання магістерського дисертаційного дослідження. Тому актуальними є питання щодо формулювання об'єкта і предмета досліджень та застосування доказових методів їх виконання. За мету цієї статті ставилося передовсім, надання рекомендації на основі фундаментальних положень інженерії програмного забезпечення щодо формулювання об'єкта і предмета магістерської дисертації зі спеціалізації в контексті інженерії програмного забезпечення. А також, вказати на застосування в магістерської дисертації наукових методів доказових досліджень і звернути увагу на оформлення результатів. Досягнення мети статті розглядається в контексті причин і умов, які історично склалися в навчанні і відповідних проблем, які виникли в навчальних планах, виборі тем і змісті досліджень магістерських дисертацій. Одна з причин, яка має об'єктивний характер, виникає в контексті сполучення домену інженерії програмного забезпечення з прикладними доменами, яке є основою відповідної спеціалізації. Стаття розрахована на магістрів зі спеціалізацій інженерії програмного забезпечення та їхніх наукових керівників.

Ключові слова: інженерія програмного забезпечення, навчання, магістерські дослідження, науково-експериментальні методи досліджень.

Вступ

Основи інженерії програмного забезпечення почали закладатися 1968 року після першої конференції, що відбулася в Гармиші (Німеччина) [1]. Вже на цій конференції було звернуто увагу на гостру необхідність підготовки кадрів, нестача яких була однією зі складових кризи програмного забезпечення, що на той час мала місце [2].

В Україні до 2006 року фахівці з інженерії програмного забезпечення не готувалися. У рамках бакалаврату комп'ютерні науки вивчали програмісти, які після закінчення університету переважно самостійно або на курсах набували додаткові знання та вміння, щоб більшменш відповідати вимогам не тільки, щодо кодувальника програм, а і щодо інженера з програмного забезпечення. На підприємствах тоді вже створеної Асоціації «Інформаційні технології України» (https://itukraine.org.ua/) відчувався гострий брак справжніх фахівців, на ринку праці їх було вкрай мало. Системного розуміння причин ситуації не було. Вважалося, що університети погано готують фахівців. Тому великі компанії самі почали підготовку кадрів.

Слід зазначити, що в Україні, в академічному середовищі ще раніше були створені передумови для вирішення проблеми в контексті інженерії програмного забезпечення. До цих передумов слід віднести роботи, які велися в Інституті кібернетики АН УРСР, здійснювалися дослідження і створювалися програмні засоби для реалізації так званої технології програмування, під якою розумілася саме інженерія програмного забезпечення [3]. Глушков В.М. підтримував ці роботи [4]. До Києва приїжджали відомі вчені, які працювали у цій галузі, як радянські, так і закордонні. Зокрема, А. Єршов, Є. Дейкстра. 1978 року проведено Першу Всесоюзну конференцію з технології програмування [5]. 1995

року в Інституті кібернетики АН УРСР була захищена перша в пострадянській Україні докторська дисертація, присвячена вирішенню актуальних на той час завдань інженерії програмного забезпечення, що належали до повторного використання програмного забезпечення [6].

2000 року в Національному авіаційному університеті було відкрито першу в Україні кафедру інженерії програмного забезпечення [7], а в Асоціації «Інформаційні технології України», до якої тоді належали тільки підприємства індустрії програмного забезпечення України, була введена позиція і обрано віце-президента для зв'язку з університетами. Як виявилося пізніше, він мав добре розуміння зв'язку проблем Асоціації з відсутністю підготовки фахівців саме з інженерії програмного забезпечення. Тоді разом зі створеною кафедрою і Асоціацією почалися роботи як у викладацькому середовищі, так і в індустрії з роз'яснення необхідності підготовки кадрів з інженерії програмного забезпечення. Таким чином було розпочато підготовку до відкриття в Україні бакалаврату з інженерії програмного забезпечення [8].

У контексті цих робіт у Національному авіаційному університеті кафедрою інженерії програмного забезпечення почав видаватися науковий журнал «Інженерія програмного забезпечення» [9], проводилася щорічна науково-технічна конференція студентів та аспірантів з однойменною назвою. Також кафедрою інженерії програмного забезпечення та Асоціацією «Інформаційні технології України» було видано публікації та проведено низку семінарів і конференцій [10 - 12].

Навесні 2006 року бакалаврат «Програмна інженерія» було відкрито, а восени здійснено перший набір студентів. До відкриття у Національному авіаційному університеті було розроблено навчальний план, на основі шаблону N2S-1c з [13]. План було прийнято на конференції завідувачів кафедр, які готували бакалаврів та фахівців із комп'ютерних наук. Ці кафедри, відповідно до рекомендації міністерства освіти були зобов'язані переходити на підготовку бакалаврів з напряму «Програмна інженерія» (інженерія програмного забезпечення). Процедура обговорення і прийняття цього плану висвітлила значні проблеми (що є і будуть в майбутньому), які пов'язувалися з відсутністю педагогічних кадрів, із розумінням, як готувати саме інженерів з програмного забезпечення.

Зараз більшість класичних та політехнічних університетів України готують бакалаврів за спеціальністю 121 – «Інженерія програмного забезпечення» та магістрів за відповідними спеціалізаціями. В Україні створено освітні стандарти для підготовки бакалаврів та магістрів [14]. На основі спеціальності 01.05.03 – «Математичне та програмне забезпечення обчислювальних машин, комплексів та мереж» створено докторантуру філософії спеціальності «Інженерія програмного забезпечення».

Але проблеми 2006 року залишилися і ненайкраще впливають на підготовку фахівців і саме магістрів.

Проблеми підготовки магістрів з інженерії програмного забезпечення

2006 року у Харкові на конференції з введення бакалаврату інженерії програмного забезпечення в Україні, групою з Києва (Національний авіаційний університет та Асоціація «Інформаційні технології України») було запропоновано навчальний план підготовки (Табл. 1, наведено англійською мовою [13]).

Учасники конференції зустріли план без ентузіазму. Пояснити це можна трьома причинами:

1. Завідувачі кафедр розуміли, що на кафедрах немає викладачів, спроможних викладати нові дисципліни за планом, і їх була переважна більшість;

2. У плані було показано різке скорочення кредитів (навчальних годин) у порівнянні із планами комп'ютерних наук, на такі дисципліни як математика, фізика, відсутня також низка дисциплін, зокрема, з електротехніки, нарисної геометрії тощо;

Таблиця1.

Software engineering, Fundamental and professional subjects

	Name of subjects	***Credits ECTS***
2.1	Mathematical Analysis	3,00
2.2	Linear Algebra and Analytic Geometry	3,00
2.3	Mathematical Statistics	3,00
2.4	Basics of Discrete Mathematics	5,00
2.5	Discrete Structures	3,50
2.6	Physics	4,00
2.7	Empirical Methods of Software Engineering	4,00
2.8	Fundamental of Programming	7,00
2.9	Object Oriented Programming	7,00
2.10	Algorithm and Data Structures	5,00
2.11	Software Construction	4,00
2.12	Software Architecture and Design	
2.13	Computer Architecture	5,00
2.14	Software Requirements Analysis	5,00
2.15	Social Safety and Labor Protection	4,00
2.16	Software economic	1,50
2.17	Software Modeling	5,00
2.18	Computer Networks	3,00
2.19	Operating Systems	6,00
2.20	Group Dynamics and Communications	2,00
2.21	Introduction to Software Engineering	7,50
2.22	Database	6,00
2.23	Human Computer Interaction	3,50
2.24	Software Quality and Testing	4,00
2.25	Program and Data Security	3,50
2.26	Professional Software Engineering Practice	2,50
2.27	Design Practice	1,50
2.28	Software Project Management	3,00
		####

3. У плані також були відсутні дисципліни, які традиційно викладалися в комп'ютерних науках, як-от, чисельні методи, основи прийняття рішень, дослідження операцій, інформаційні технології і т. і.

Проте групі з Києва вдалося знайти підтримку деяких завідувачів кафедр, і запропонований план було взято за основу. Пізніше цей план було рекомендовано Навчально-методичною підкомісією до застосування в стандарті при відкритті бакалаврату «Програмна інженерія».

Але, позаяк, план мав рекомендаційний характер, то і застосовували його в повному обсязі тільки окремі університети, наприклад, Національний авіаційний університет. На жаль, пізніше, коли почали прийматися стандарти навчання як першого, так і другого рівня в них були відсутні навчальні плани. Їх опосередковано представляли переліки компетенцій, вмінь та результатів навчання і освітні програми. Реалізувати їх можна було формально, шляхом створення будь-якого плану, зручного кафедрі. І зараз, зазвичай, ці плани найчастіше віддалено пов'язані з інженерією програмного забезпечення. Пояснюється це тим, що стан з відсутністю компетентних викладачів, на жаль, не змінився.

Наразі залишаються умови, через які виникли проблеми і з підготовкою магістрів. Проблеми виявилися знову, ще і в навчальних планах і виборі тем, і в змісті досліджень магістерських дисертацій. Ситуація, що склалася, має три причини. Дві є наступними (про третю трохи згодом):

1. Підготовка бакалаврів тільки частково відповідає спеціальності;

2. Некомпетентність в інженерії програмного забезпечення викладачів–наукових керівників магістрів.

Як бачимо, обидві причини сягають корінням у минуле. Наслідком першої причини є неготовність студентів до грамотного розв'язання завдань інженерії програмного забезпечення. Наслідком другої причини є вибір тем магістерських досліджень, далеких від інженерії програмного забезпечення. Ці ж причини «працюють» у бакалаврських дипломних роботах. Наприклад, типові теми для бакалаврських робіт - це будь який Web-сайт, зокрема, засоби моніторингу продажу квитків, контролю артеріального тиску, серцево-судинної системи, розташування сервісів у місті тощо.

Розглядаючи навчання магістрів, звернемо увагу на питання вибору об'єкта та предмета дослідження, а також

на структуру та зміст дисертації. Сподіваємося, що відповіді на ці питання, які пропонуються в статті, допоможуть науковим керівникам і студентам відповідних спеціалізацій Інженерії програмного забезпечення.

Аналіз літератури

Усі праці, близькі за темою цій статті, можливо поділити на три групи.

До першої групи віднесемо праці, які стосуються становлення навчання інженерії програмного забезпечення як окремої спеціальності [14 - 19]. До другої групи - праці, які стосуються змісту навчання, саме інженерії програмного забезпечення як, скажімо, окремого напряму підготовки (бакалаврату) [20 - 27]. До третьої групи віднесемо праці, що стосуються підготовки магістрів [28 - 34]. Зрозуміло, що найбільш близькими до теми цієї статті є праці третьої групи.

Зміст праць першої групи найчастіше зосереджувався на ознаках інженерії програмного забезпечення як інженерної діяльності [17-19]. Ознаки зазвичай показувалися на прикладі відмінності інженерії програмного забезпечення від комп'ютерних наук [15]. Це пов'язано с відокремленням свого часу, інженерії програмного забезпечення як освітньої і виробничої спеціальності саме від комп'ютерних наук.

Визначається, що, по-перше, ці дві галузі діяльності відрізняються за методом творчого мислення. Для розв'язання завдань комп'ютерних наук застосовується метод конвергентного мислення, коли для розв'язання передбачаються точні (часто математичні) інструкції, тож мається на увазі, що існує єдине рішення. Це вказує на те, що задачі комп'ютерних наук мають переважно теоретичне підґрунтя. Для вирішення завдань інженерії програмного забезпечення застосовується метод дивергентного мислення, коли для розв'язання пропонується декілька рішень. Серед рішень обирається одне й до його реалізації застосовується метод конвергентного мислення. Водночас на вибір єдиного рішення впливають чимало факторів, більшість з яких пов'язані а практичним підґрунтям розв′язуваного завдання.

По-друге, результат розв'язання завдання інженерії програмного забезпечення завжди є продуктом, у якого є користувач. Продукт створюється в контексті життєвого циклу і має задовільняти низку вимог, характерних для інженерії (конструкція продукту, якість, стандарти, документованість, супровід, вплив на зовнішнє середовище). Одночасно, частина робіт першої групи присвячена з'ясуванню поняття «інженер із програмного забезпечення» [17, 18].

Праці другої групи насамперед було присвячено з'ясуванню навчання як переходу від окремої або декількох дисциплін до системно сформованого переліку дисциплін, спрямованого на навчання спеціальності за відповідним змістом. Дискусії навколо змісту навчання засновані на обговоренні або використанні SWEBOK [21]. Для побудови навчальних планів і програм переважна більшість університетів почала застосовувати рекомендації SE2004 [14].

На сьогодні інженерія програмного забезпечення - це систематизований, регламентований і квантифікований підхід до реалізації процесів створення, експлуатації, супроводу та утилізації програмного забезпечення. До того ж процеси, ресурси та програмні продукти інженерії програмного забезпечення мають відповідати заданим технічним, економічним, соціальним, правовим вимогам, а також вимогам стійкого розвитку, зокрема, «зеленим» [35].

Праці третьої групи спрямовані на підготовку з інженерії програмного забезпечення з урахуванням існуючих практик - прикладних доменів [28-33]. Хоча дослідження в інженерії програмного забезпечення, безумовно, також потребують підготовки магістрів, саме наявність прикладних доменів враховується в підготовці магістрів переважною кількістю університетів, і вона приводить до появи відповідних спеціалізацій спеціальності Інженерія програмного забезпечення. Водночас існують роботи, де ці практики класифіковані, зокрема,

шляхом аналізу понад десяти тисяч виконаних проєктів [31]. Подібні класифікації забезпечують аргументовану базу для вибору спеціалізацій щодо підготовки магістрів. Але, як показує аналіз і досвід автора, наявність прикладних доменів часто призводить до абсолютного зсуву акцентів досліджень магістерських дисертацій з об'єктів і предметів інженерії програмного забезпечення в прикладні домени. Тому зауважимо, що саме зсув змісту навчання магістрів і, зокрема, досліджень магістерських дисертацій з інженерії програмного забезпечення на окремі прикладні домени є третьою причиною появи питань, які розглядаються в статті. Таким чином, до проблем підготовки студентів і некомпетентності викладачів додається проблема сполучення домену інженерії програмного забезпечення з прикладними доменами, наприклад [29], військовим – застосунки для військових; з доменами систем – застосунки для управління пристроями; комерційним – застосунки для обробки інформації різноманітного представлення (текстового, графічного, табличного, планування ресурсів тощо); аутсорсинговим доменам – застосунки, які виконуються, мають відмінності в зв'язку з контрактними відношеннями; інформаційних систем – застосунки для управління бізнес функціями.

Разом с тим, огляд літератури показує, що в умовах, які склались, відсутні роботи щодо особливостей підготовки магістерських дисертацій з інженерії програмного забезпечення в умовах, які склалися. Водночас, якщо перши дві причини проблем підготовки магістрів можна і потрібно усувати, то третя причина - об'єктивна і її треба враховувати під час навчання магістрів.

Це стосується визначення об'єкта й предмета дисертації, назви, змісту та опису досліджень і результатів з урахуванням наявності прикладного домену. На відміну від закордонних університетів, в Україні протягом багатьох років склалася й існує практика формулювання цілей дослідження, застосовуючи поняття об'єкта і предмета дослідження. І це, як показав час, важливий етап, від виконання якого залежить результативність дослідження.

Магістерська дисертація. Об'єкт та предмет

Основи наукових досліджень вчать, що, перш ніж братися безпосередньо за виконання досліджень необхідно визначити об'єкт і предмет дослідження.

Об'єкт дослідження визначається шляхом встановлення найбільш загального явища в домені, в контексті якого буде виконуватися дослідження. Це даність, яка легко встановлюється, якщо у дослідника достатньо знань про сутність дослідження і досліджуваного домену. Для спеціалізацій, що розглядаються, це знання з домену - Інженерія програмного забезпечення.

Предмет дослідження належить до одного або декількох аспектів явища, яке визначено в якості об'єкта дослідження і визначається метою дослідження, на основі особливостей прикладного домену. Для спеціалізацій, що розглядаються, цей домен може бути, зокрема, наступним - Інформаційні системи і обробка даних (SAS.inf Information systems and data processing [13]).

Правильність вибору об'єкта і предмета дослідження впливає з одного боку на результативність дослідження, а з іншого - на можливість правильної оцінки отриманих результатів.

Якщо об'єкт і(або) предмет дослідження обрано невірно, тоді мають місце два випадки.

У першому випадку виявляється, що отримані результати аж ніяк не є тими, що мають бути отримані в досліджуваному домені.

У другому випадку, експерти (екзаменаційна комісія), увага яких спрямовуватиметься на оцінку результатів дослідження, можуть виявитися не спеціалістами в домені, в якому, за припущенням дослідника, проводилося дослідження. Як наслідок - оцінити результати експерти не зможуть.

Зв'язок об'єкта і предмета мусить бути безпосереднім і вимагати, аби

Рисунок 1. Життєвий цикл

об'єкт і предмет були на перетині доменів спеціалізації, а саме, як-от Інженерія програмного забезпечення й Інформаційні системи і обробка даних. Запропонуємо рекомендації щодо визначення цього зв'язку.

Для визначення зв'язку між об'єктом і предметом необхідна інформація для уточнення поняття об'єкта дослідження. Будемо це робити, використовуючи поняття життєвого циклу як системоутворюючого в інженерії програмного забезпечення (Рис.1).

Одночасно, використовуємо те, що всі фази життєвого циклу мають три складові - продукт, процес і ресурс (Рис. 2).

Таким чином, конкретизуючи можливий об'єкт дослідження, будемо розглядати наступне (рис.3): продукти (робочі продукти - результати виконання процесів фаз життєвого циклу; програмні продукти - результати виконання життєвого циклу при створенні програмного забезпечення); процеси (процеси фаз життєвого циклу); ресурси, розділяючи їх на дві частини, ресурси I - програмно-реалізовані ресурси (інструменти і артефакти, наприклад, тести, представлення стилів, документи) і ресурси II. До ресурсів II віднесемо підходи і методи, використовувані в інженерії програмного забезпечення. Наприклад, підхід компонентного створення і супроводження, метод Scram, щодо організації команд (організацій) за методологію Agile. В процесі конкретизації обов'язково маємо на увазі наступні розділи інженерії програмного забезпечення: різновид інженерії – пряма, зворотна, емпірична (метричне забезпечення); горизонтальні процеси і відповідні продукти - документування, валідація, верифікування, керування, гарантування якості; економіка програмного забезпечення; зелене програмне забезпечення і сталий розвиток; культура інженерії програмного забезпечення і правові питання; екосистеми програмного забезпечення.

Наведені уточнення об'єкта дослідження слід розглядати в контексті чотирьох під доменів спеціалізації інженерії програмного забезпечення:

- прикладному, для якого створюється програмне забезпечення (програмний продукт). Наприклад, згідно з спеціалізацією, це інформаційні технології (представлена відповідними знаннями XBoK, як-от, для інформаційних систем і технологій це CBoK [36]);

- проблемному, в якому розв'язуються нові, чи удосконалюються рі-

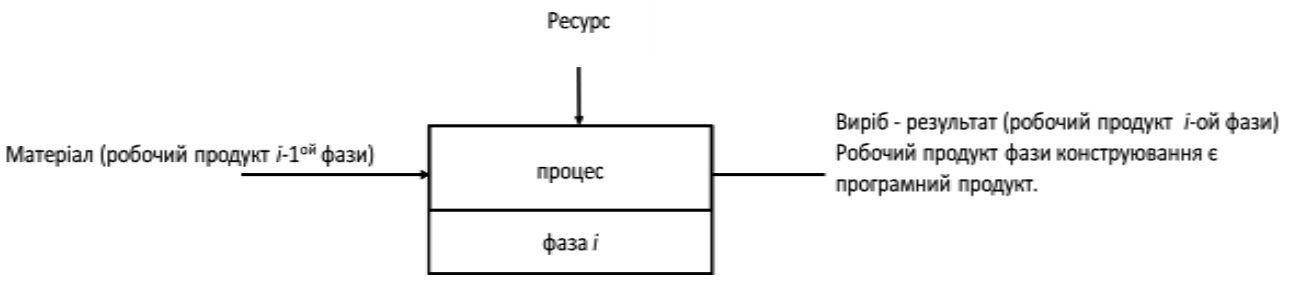

Рисунок 2. Складові фаз життєвого циклу

Рисунок 3. Об'єкт дослідження. Класифікація

шення існуючих, наукові завдання щодо створення і супроводження програмного забезпечення (програмний продукт) - наукова частина домену інженерії програмного забезпечення (представлена SWEBoK [21]);

- реалізаційному, в якому створюється програмне забезпечення (програмний продукт) – інженерна частина домену інженерії програмного забезпечення (представлена SWEBoK);

- домену оцінювання і впровадження результатів, де перевіряються окремі властивості отриманих результатів та досліджуються питання їх впровадження – практична частина домену інженерії програмного забезпечення.

Зв'язки цих доменів визначають структуру магістерського дослідження (рис. 4).

Таким чином, для формулювання об'єкта, предмета та назви дисертації необхідно виконати наступні три кроки (рис. 5):

Аналізуючи завдання дослідження, яке сформульовано в прикладному домені, обираємо в домені інженерії програмного забезпечення об'єкт дослідження,

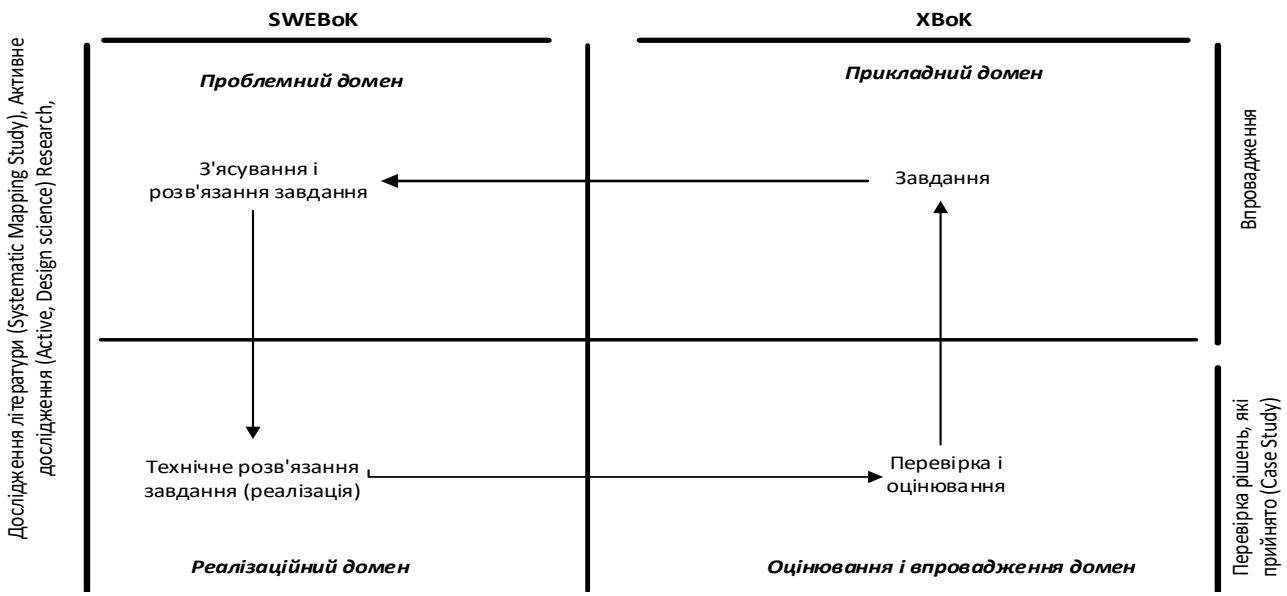

Рисунок 4. Структура магістерського дослідження

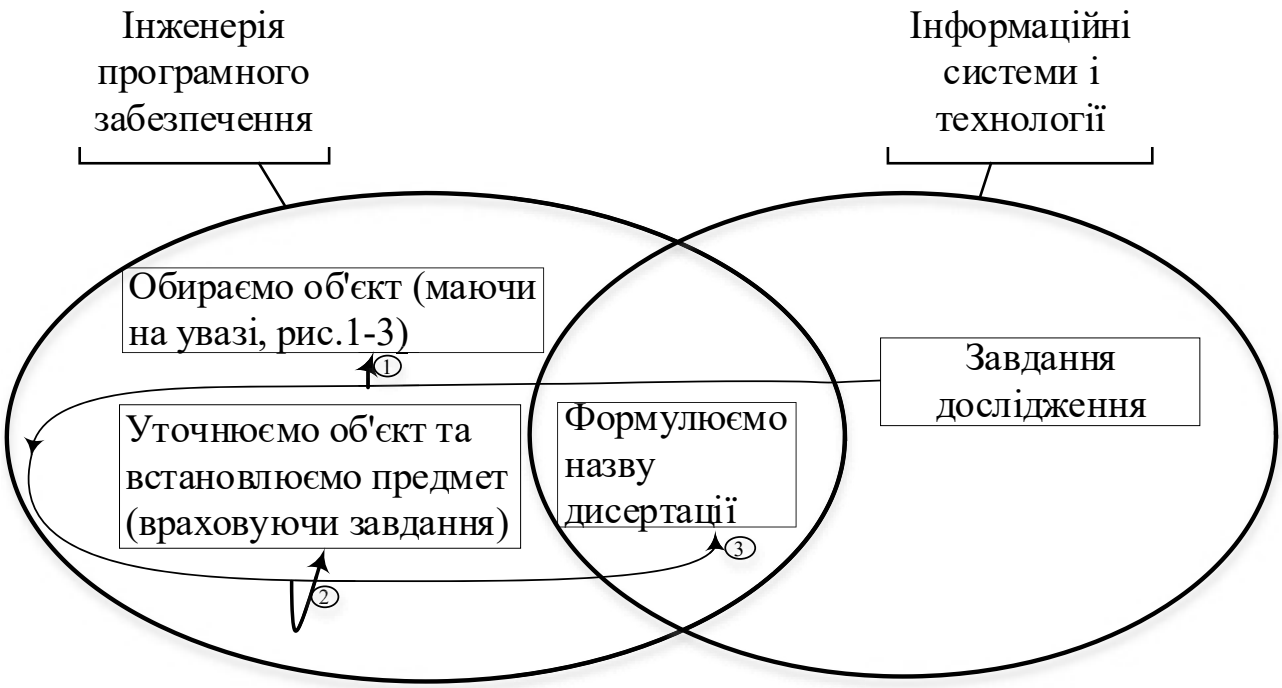

Рисунок 5. Зв’язок завдання-об’єкт-предмет-назва дисертації

спираючись на класифікацію, наведену на рис. 1-3.

Продовжуючи аналіз завдання, уточнюємо (якщо потрібно, шляхом виконання розвідувального дослідження літератури, рис. 6) об’єкт дослідження та встановлюємо предмет дослідження, який визначається об’єктом та цілями (завданням) дослідження. Як правило, це методології, методи, моделі і засоби створення і супроводу програмного забезпечення або відповідне уточнення (продукти, процеси, ресурси) згідно із наведеними рисунками 1-3 для прикладного домену.

Нарешті формулюємо назву дисертації. Здебільшого назва дисертації співпадає з формулою предмета дослідження.

Після виконання цих кроків можна починати магістерське дослідження згідно із структурою, яку наведено на рис. 4.

Застосування рекомендацій

Розглянемо такий випадок. Спеціалізація - інженерія програмного забез-

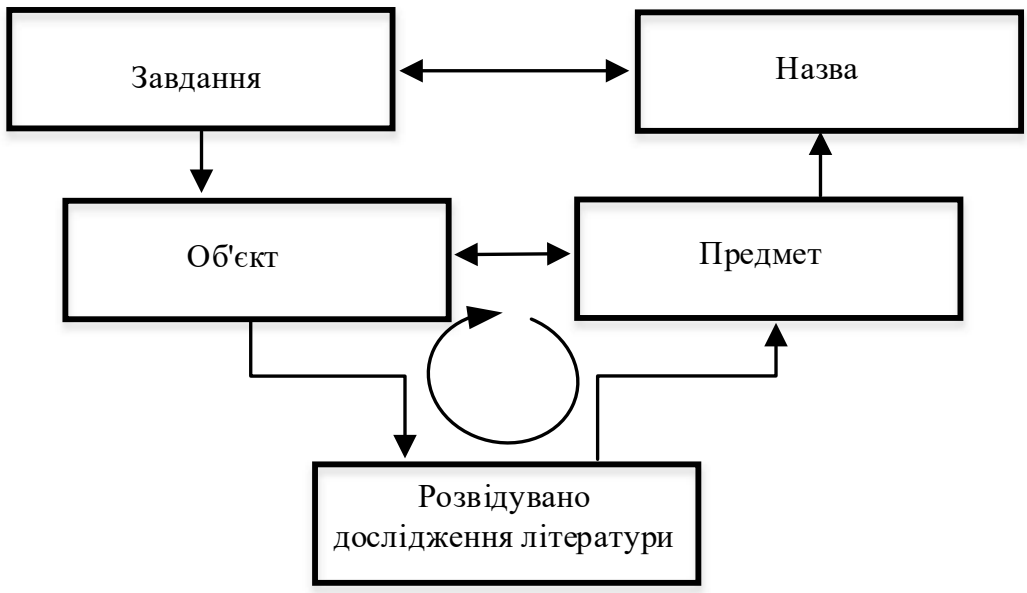

Рисунок 6. Уточнення об’єкта та встановлення предмета

печення інформаційних систем і технологій, домен - інженерія програмного забезпечення, прикладний домен – інформаційні системи і технології. Завдання – створити і дослідити Web - сервіс, що забезпечує прийняття рішень на основі неповної інформації.

Досвід показує, що зазвичай за об'єкт дослідження обирають програмне забезпечення Web – сервісу.

В цьому випадку предметом дослідження визначають метод і алгоритми прийняття рішень на основі неповної інформації (незалежно від того, існують вони чи ні), які необхідно встановити і реалізувати в рамках програмного забезпечення Web - сервісу.

Отож, якщо об'єкт дослідження визначено майже відповідно до домену, то предмет не має ніякого відношення, ні до інженерії програмного забезпечення, ні до перетину цього домену і прикладного домену. Метод і алгоритми прийняття рішень на основі неповної інформації як предмет дослідження цілком належить до домену інформаційних систем і технологій.

Таким чином, отримані результати (метод і алгоритми прийняття рішень на основі неповної інформації) будуть тільки частково належати до спеціалізації, тому експерти з цієї спеціалізації не зможуть їх визначити. Потрібні також експерти з іншого домену, а саме з домену інформаційних систем і технологій. Хоча дослідник повністю переконаний у правильності визначення об'єктом програмне забезпечення, предмет дослідження може бути з ним і не пов'язаний.

Скористаємося вище згаданими кроками (рис.5, 6). На виконання кроку 1 об'єктом дослідження для наведеного випадку (беручи до уваги, що Web – сервіс реалізується шляхом створення відповідного програмного забезпечення, а інших уточнень у завданні немає) може залишатися програмне забезпечення Web - сервісу, що уможливлює прийняття рішень на основі неповної інформації, або відповідне уточнення згідно з наведеними рисунками 1-3.

На другому кроці (рис. 5) цілями (завданням) дослідження визначається предмет дослідження. Як правило, це методології, методи, моделі й засоби створення й супроводу програмного забезпечення, або відповідне уточнення (якщо є в завданні). А саме продукти, процеси, ресурси відповідно до наведених рисунків 1-3. Наприклад, якщо програмне забезпечення розглядається в контексті прийняття рішень на основі неповної інформації, предметом може бути наступне: компоненти неодноразового використання (необов'язково програмні) відповідної форми (робочий продукт фази доменного аналізу) для створення програмного забезпечення Web - сервісу, або процес(и) (фаза(и) життєвого циклу) створення і супроводження програмного забезпечення. Або ж представлення і перетворення знань для створення і супроводження програмного забезпечення (ресурс I – артефакт) або підхід (метод) для створення і супроводження програмного забезпечення (ресурс II) .

В такому разі перевизначимо предмет дослідження наведеного раніше прикладу наступним чином, - методи (і, можливо, засіб), моделі, інструменти створення (супроводження) програмного забезпечення Web - сервісу, що забезпечує реалізацію методу прийняття рішень на основі неповної інформації.

Можливі й інші визначення предмету дослідження залежно від уточнень, які можуть бути в завданні:

- методи, моделі, інструменти створення і супроводження компонентів неодноразового використання для Web - сервісу, що забезпечує прийняття рішень на основі неповної інформації;
- методи, моделі, інструменти для реалізації процесу(ів) створення і супроводження програмного забезпечення Web - сервісу, що уможливлює прийняття рішень на основі неповної інформації;
- методи, моделі, інструменти для представлення і перетворення знань щодо програмного забезпечення Web - сервісу, яке гарантує прийняття рішень на основі неповної інформації.

Відповідно, предметом дослідження буде не метод і алгоритми прийняття рішень на основі неповної інформації, як

це визначено раніше, а метод створення (далі за об'єктами наведених прикладів) програмного забезпечення Web - сервісу, в якому застосовуються метод і алгоритми прийняття рішень на основі неповної інформації.

Як бачимо, об'єкт дослідження - це програмне забезпечення, а, визначаючи предмет, уточнюємо - робочий продукт, процес, або артефакт для прикладного домену - інформаційні системи і технології, де передбачається використання певних методів і алгоритмів прийняття рішень. Водночас використання методу і алгоритмів потребує наступного: розробки нового, або суттєвого удосконалення існуючого підходу до створення або супроводження програмного забезпечення; створення і супроводження компонентів неодноразового використання; реалізації процесу(ів) створення і супроводження програмного забезпечення; створення інструментів для представлення і перетворення знань щодо програмного забезпечення.

Звісно, використання методу і алгоритмів прийняття рішень у рамках програмного забезпечення Web - сервісу має вимагати розв'язання певного науково-практичного завдання, що обов'язково належить до інтересів інженерії програмного забезпечення. Для розв'язання цього завдання в процесі дослідження і буде створюватися метод та, ймовірно, засіб. Саме ці метод і засіб мають бути результатами дисертаційного дослідження, а не програмне забезпечення Web - сервісу, що забезпечує прийняття рішень на основі неповної інформації.

Виконуючи крок 3, беремо до уваги, що визначення предмету дослідження практично завжди буде назвою дисертації.

Так, зокрема, маємо наступні назви:

Метод і засіб створення і супроводження компонентів неодноразового використання щодо розробки і супроводження програмного забезпечення Web - сервісу, який забезпечує прийняття рішень на основі неповної інформації.

Метод і моделі щодо реалізації процесу специфікування вимог програмного забезпечення Web - сервісу, який забезпечує прийняття рішень на основі неповної інформації.

Моделі і інструменти для представлення і перетворення знань щодо створення і супроводження програмного забезпечення Web - сервісу, яке уможливлює прийняття рішень на основі неповної інформації.

Магістерська дисертація. Зміст

Після визначення об'єкта і предмета дослідження можна виконувати дослідження. Магістерська дисертація як основне дослідження (звичайно, це action research [15]) має наступні складові:

- розвідувальне дослідження для з'ясування об'єкта та предмета основного дослідження. Як правило, його не буває, якщо досвіду наукового керівника і знань студента достатньо. Але через умови, про які йшла мова раніше, це дослідження необхідно проводити;
- вивчення стану об'єкта (предмета), шляхом дослідження літератури. Міститься в першому розділі дисертації;
- основне (первинне) дослідження (action research) – дослідження предмета і створення теорії, підходу, методу, засобу, моделі або іншого для вирішення практичної задачі (відповідно назви дисертації). Міститься в другому та третьому розділах дисертації;
- дослідження теорії, підходу, методу, засобу, моделі або іншого, які створено в основному дослідженні (вторинне дослідження). В деяких випадках може сполучатися з контрольованим експериментом. Міститься в четвертому розділі дисертації.

Отож, зазвичай, текст магістерської дисертації описує результати трьох досліджень і містить чотири розділи.

З'ясування та вивчення стану об'єкта і предмета здійснюються шляхом дослідження літератури. Для виконання цих досліджень в інженерії програмного забезпечення найчастіше застосовують метод Систематичного огляду літератури - Systematic mapping study [37]. Це метод вторинного дослідження,

яке використовує чітко визначену методологію для виявлення, аналізу та інтерпретації всіх наявних доказів, пов'язаних із конкретним, первинним дослідженням, тож, є неупередженим та (певною мірою) повторюваним, а тому доказовим.

Основними етапами процесу дослідження за методом Systematic mapping study є визначення дослідницьких питань, проведення пошуку відповідних джерел, відбір джерел, дослідження анотацій та безпосередньо текстів, котрі знайдено за ключовими словами, збір і представлення даних [37]. Кожен етап процесу має результат. Кінцевим є графічне представлення результатів дослідження та обговорення. Основною метою Systematic mapping study є огляд літератури щодо досліджуваної галузі, визначення кількості та типу досліджень, а також наявних у галузі результатів. Крім цього, існує ще дві цілі досліджень:

- перша - відображення частоти публікацій і їх хронології, щоб побачити тенденції досліджень;
- друга - визначення джерел, на які спиралися дослідження в цій галузі.

Для візуалізації результатів зазвичай застосовують діаграму бульбашок з MS Excel.

У частині розділу – обговорення стану досліджень предмету дисертації з повним та інтенсивним застосуванням посилань на знайдені джерела. Після чого формуються підстави для проведення первинного (основного) дослідження.

Первинне дослідження (action research) – дослідження предмета і створення теорії, підходу, методу, засобу, моделі тощо для вирішення науково-практичної задачі, яку поставлено в дисертації.

Активне (дієве) дослідження (Action research) – це дослідження, з метою розв'язання науково-практичної задачі науковими методами, шляхом впливу на явище з будь-якого аспекту або зміною явища, що є фокусом дослідження [38]. Рішення задачі здебільшого потребує вивчення відповідного явища в аспекті предмета дослідження. Тож, активне (дієве) дослідження поєднує наукові методи з практичними задачами, які потребують розв'язання. Водночас дослідник повинен втручатися (впливати, діяти) в явище, змінюючи і перетворюючи його або його контекст. Активне дослідження має глибокий гносеологічній характер, тому, на жаль відсутні методики його проведення. Тут важливу роль відіграє науковий керівник здобувача. Саме від нього залежить, чи буде досягнута мета, чи будуть відповідати результати активного дослідження темі дисертації. Відповідність буде перевірено шляхом виконання вторинного дослідження.

Вторинне дослідження, метод дослідження результату (action research - Case Study). Case study - це метод виконання наукових досліджень, зокрема, в інженерії програмного забезпечення, невеликих явищ. Здебільшого застосовується для обґрунтування результатів первинного дослідження наукових статей, бакалаврських робіт, магістерських дисертацій. Тому засвоєння цього методу і застосування в магістерської дисертації додає дисертації наукового, ґрунтовного і доказового характеру. Case Study - це метод досліджень результату Action research, для розуміння та пояснення явища чи перевірки теорії, методу або засобу, отриманих у результаті виконання основного дослідження, використовуючи переважно (але необов'язково) якісний аналіз [39]. Case Study - це емпіричний метод дослідження. Це не підмножина, або варіант інших методів, таких як, контрольований експеримент, огляд чи історичне дослідження [40]. Case Study спрямоване щодо об'єкта та предмета основного дослідження на наступне:

1. Детально відповідає на питання «як і навіщо»;
2. Забезпечує глибоке розуміння причин і наслідків основного дослідження;
3. Забезпечує тестування теорій і методів, засобів, запропонованих у первинному дослідженні, коли змінні недостатньо керовані.

Існують наступні типи Case Study [39]:

- пояснювальний – досліджує і пояснює «як і чому так?». Наприклад, показує різницю між теоріями, підходами, методами або сутність теорії, підходу, ме-

тоду, або переваги теорії, підходу, методу, які створено;

- описовий – досліджує і показує «як це діє?» - можна віднести до засобів, які створено;

- причинний – досліджує і показує «яким чином?». Наприклад, як створений стиль кодування впливає на розуміння вихідного коду. Цей тип можна визначати шляхом оцінювання ефективності отриманих результатів;

- розвідувальний (попередній) – досліджує і показує «який стан?». Наприклад, дослідження літератури для з'ясування стану дослідження об'єкта або предмета основного дослідження, зокрема, за методом Systematic mapping study.

Може бути Case study змішаного типу. Case study є добре регламентованим методом. Існують докладні методики щодо виконання того чи іншого типу Case study [39].

Case study також має об'єкт та предмет, які формулюють у контексті результатів основного дослідження. Наприклад, основне дослідження – Метод і засоби редокументування успадкованого програмного забезпечення; Основне дослідження – об'єкт - успадковане програмне забезпечення; предмет – процеси, моделі, методи, засоби редокументування успадкованого програмного забезпечення. Тоді Case study - об'єкт - процеси, моделі, методи, засоби редокументування успадкованого програмного забезпечення; предмет – метод і засіб редокументування успадкованого програмного забезпечення, розроблені в основному дослідженні. Питання Case study – Чи працездатні та ефективні метод і засіб редокументування успадкованого програмного забезпечення, створені в основному дослідженні?

У виконанні досліджень щодо інженерії програмного забезпечення дуже важливо застосування вимірювань [40]. Для цього існують метрики і відповідні вимірювачі. Але трапляються випадки, найчастіше саме у проведенні основних досліджень і Case study, коли потрібних метрик або не існує або не зрозуміло які метрики слід застосовувати в конкретному випадку. Тоді використовують широко відомий в інженерії програмного забезпечення метод ціль-питання-метрика - GQM [41].

Метод GQM базується на припущенні, що для цілеспрямованого вимірювання, спочатку необхідно вказати ціль, а далі простежити цю ціль до метрик, які призначені для кількісного визначення цілі і, зрештою, забезпечити основу для інтерпретації метрик щодо заявленої цілі. Для полегшення простеження цілі до метрик застосовують питання, які пояснюють ціль. Таким чином модель GQM має три рівні: концептуальний (ціль); операційний (питання); кількісний (метрики).

Концептуальний рівень (Goal) визначає мету для об'єкта вимірювань з різних причин і точок зору, щодо конкретного середовища. Об'єкти вимірювання такі: продукти, процеси, ресурси.

Операційний рівень (Question) визначає набір питань, які використовуються для характеристики способу оцінки/досягнення конкретної мети. Формулювання питань має здійснюватися на основі характеристик об'єкта. Питаннями намагаються охарактеризувати об'єкт вимірювання (продукт, процес, ресурс) і визначити його з обраної точки зору.

Кількісний рівень (Metric) - це набір даних, метрика. Пов'язаний з кожним запитанням для відповіді на них кількісно. Дані можуть бути наступними:

- об'єктивні - якщо вони залежать лише від об'єкта, властивість якого вимірюється, а не від точки зору. Наприклад, кількість версій документа, кількість годин, витрачених на виконання завдання, розмір програми.

- суб'єктивні - якщо вони залежать і від об'єкта, властивість якого вимірюється, і від точки зору. Скажімо, читабельність тексту, рівень задоволеності користувачів.

Наприклад, об'єкт – стандарт кодування (ресурс), призначення – підвищення ефективності кодування, фокус – програмування, точка зору – менеджера, середовище – мова програмування.

Оформлення результатів (Good papers). Результати досліджень пред-

ставляються відповідними дослідницькими текстами. Найпоширенішими формами є стаття та дисертація. В інженерії програмного забезпечення дослідницькі тексти є звичними засобами для донесення науковій спільноті результатів дослідження [42]. У дослідницькому тексті автор пояснює зацікавленим читачам, як і яких результатів він досягнув, чому читачу слід читати цей текст. Якісний дослідницький документ має відповісти на ряд питань [42]:

- Яким саме є ваш результат?
- На яке запитання ви відповіли (предмет дослідження) ?
- Чому для читача важливий ваш результат?
- У контексті якого ще більшого питання поставлено питання, на яке ви відповіли (об'єкт дослідження)?

Щодо отриманого результату слід дати відповіді на наступні запитання:

- Які нові знання вашого результату читач може використати?
- Яку попередню роботу (вашу чи чужу) ви намагаєтесь вдосконалити?
- Чим ваш результат відрізняється і є кращим за цю попередню роботу?
- Яким конкретно є ваш новий результат?
- Чому читач має вірити вашому результату?
- Який стандарт (методику) слід використовувати для оцінки вашого результату?
-Які конкретні докази свідчать, що ваш результат задовольняє вашу вимогу?

Якщо автор чітко відповідає на ці запитання, він, імовірно, добре усвідомлює і може донести до читача свій результат. Якщо до того ж результат цікавий, зрілий та вагомий для прикладного домена в контексті інженерії програмного забезпечення, то це хороший шанс для позитивної ухвали на захисті дисертації.

Висновки

Метою цієї статті було узагальнення базових положень щодо формулювання об'єкта і предмета магістерської дисертації зі спеціалізації інженерії програмного забезпечення в умовах які склалися в Україні. По-друге, надання рекомендацій щодо застосування в магістерській дисертації наукових методів досліджень і оформлення результатів.

На відміну від закордонних університетів, в Україні протягом багатьох років склалася практика формулювання цілей дослідження, застосовуючи поняття об'єкта і предмета дослідження. І це, як показав час, важливий етап, від виконання якого залежить результативність дослідження. Загальне й тотальне проникнення програмного забезпечення в життя, з одного боку, і поява спеціалізацій інженерії програмного забезпечення - з іншого, значно ускладнюють виконання цього етапу дисертаційного дослідження. Тому актуальними є питання щодо формулювання об'єкта і предмета досліджень та застосування доказових методів їх виконання.

Отож, розглянуті умови, в яких за яких в Украйні здійснюється навчання магістрів, наведено рекомендації щодо формулювання об'єкта та предмета дисертаційного дослідження на основі фундаментальних положень інженерії програмного забезпечення, розглянуто методи доказових досліджень.

Література

1. Report on a conference sponsored by the NATO science committee, Garmisch, Germany, 7th to 11th October 1968, Editors: Peter Naur and Brian Randell.
2. Boehm B., 2006, A View of 20th and 21st Century Software engineering [Text]. ICSE'06. May 20–28. China. 2006. P. 12–29.
3. Вельбицкий И.В. Технология программирования. [Текст]. Техника. Киев. Украина. 1984. 279 с.
4. Глушков В.М., Вельбицкий И.В. Технология программирования и проблемы ее автоматизации. [Текст]. УСИМ. Киев. № 6. 1976. С. 75–93.
5. Технология программирования: Тез. докл. 1-й Всесоюз. конф., Киев, октябрь 1978 г. Пленарные докл. И общ. материалы. – Киев: Ин-т кибернетики АН УССР, 1979.
6. Сидоров М.О. Інженерія утилізації програмного забезпечення систем управлін-

ня та ЕОМ.- Автор. Дис.на здобуття наук. ступ. докт. техн. наук..- Інститут кібернетики імені В.М.Глушкова НАН України.- 1995 р. 25 с.
7. Сидоров М.О. Кафедра інженерії програмного забезпечення. Компьютер-Клас! 2001. № 8. С. 17–19.
8. Бондаренко М., Сидоров М., Морозова Т., Мендзебровський І. Модель випускника бакалаврату «Програмна інженерія», Вища школа. 2009. № 4. С. 50–61.
9. https://nau.edu.ua/ua/menu/science/faxovi-vidannya/
10. Сидоров Н.А. Инженерия программного обеспечения – дисциплина или бакалаврат? Матеріали міжнародної науково-практичної конференції «Розробка систем програмного забезпечення: виклики часу та роль інформаційному суспільстві». Київ, 27-28 січня 2005 р.
11. Сидоров Н.А. Инженерия ПО - дисциплина или бакалаврат? Корпоративные системы. № 2. 2005. С. 22–27.
12. Сидоров Н.А. Инженерия программного обеспечения – учебная дисциплина или подготовка бакалавра? Управляющие системы и машины. 2006. № 2. С. 25–34.
13. Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering A Volume of the Computing Curricula Series, August 23, 2004
14. Накази Міністерства освіти і науки України від 29.10.2018 № 1166, 17.11.2020 р. № 1424
15. Parnas D. Software Engineering Programmes are not Computer Science Programmes// Annals of Software Engineering.- April 9, 1998, P1-16.
16. Boehm B. k "The IEEE-ACM Initiative on Software Engineering as a Profession", IEEE Computer Society Software Engineering Technical Council Newsletter, Vol. 13, No. 1, Sept. 1994, page 1.
17. Cheston, G.A. and Tremblay, J., Integrating software engineering in introductory computing courses. IEEE Softw.,2002, 19(5) 64–71.
18. Michael Davis Defining "Engineer" — How to Do It, and Why It Matters/Journal of Engineering Education, Vol. 85, No. 2. 1996
19. Gary Ford, Norman Gibbs, A Mature Profession of Software Engineering/Technical Report CMU/SEI-96-TR-004 ESC-TR-96-004 September 1996
20. George Hurlburt and Jeffrey Voas, Software Is Driving Software Engineering?/ IEEE Software, January/February 2016, 101-104p
21. P. Bourque and R.E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, ver. 3.0, IEEE CS, 2014; www.swebok.org.
22. https://www.acm.org/binaries/content/assets/education/se2014.pdf
23. A. Mishra et all, Software engineering education: Some important dimensions/European Journal of Engineering Education · June 2007
24. Dart, P., Johnston, L., Schmidt, C. and Sonenberg, L., Developing an accredited software engineering program. IEEE Softw., 1997, 14(6), 66–70.
25. Ford, G.A., Progress Report on Undergraduate Software Engineering Education. CMU/SEI-94-TR-11. Software Engineering Institute, Carnegie Mellon University, 1994.
26. Moore, M.M., Software engineering education. IEEE Softw., 2002, 19(5), 103.
27. Saiedian, H., Software engineering education and training for the next millennium. J. Syst. Softw., 1999, 49, 113–115.
28. Current State of Software Engineering Master's Degree Programs In the United States Donald J. Bagert and Xiaoyan Mu in Proceedings - Frontiers in Education Conference · November 2005
29. V.R. Basilli, "Software Engineering: State of the Art and Practice," Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research, R.W. Selby, ed., John Wiley & Sons, 2007, ch. 8.
30. Capers Jones, Variations in Software Development Practices, November/December 2003 IEEE SOFTWARE
31. C. Landwehr et al. Software Systems Engineering programmes a capability approach/ The Journal of Systems and Software 125 (2017) 354–364
32. A. Mishra et all, Industry Oriented Advanced Software Engineering Education Curriculum, Croatian Journal of Education Vol: 14 (3/2012), pages: 595-624
33. C. Wohlin *, B. Regnell, Strategies for industrial relevance in software engineering

education, The Journal of Systems and Software 49 (1999) 125±134
34. Graduate Software Engineering 2009(GSwE2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering, 2009, Stevens Institute of Technology
35. Н. А Сидоров, Экология программного обеспечения, Інженерія програмного забезпечення, Вип.1, 2010, С53-61
36. The ACS Core Body of Knowledge for ICT Professionals (CBOK), Australian Computer Society Inc, Sydney NSW 2000
37. K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, Proceedings of the .-12th International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 1-10.
38. Paulo Sérgio Medeiros Dos Santos, Guilherme Horta Travassos Action Research Can Swing the Balance in Experimental Software Engineering.- Advances in Computers · December 2011, P78.
39. P. Runeson, M. Case study research in software engineering, Guidelines and Examples.- 2012 by John Wiley & Sons, Inc, P241.
40. Forrest Shull • Janice Singer • Dag I.K. Sjøberg Guide to Advanced Empirical Software Engineering.- Springer-Verlag London Limited 2008, P394.
41. Basili,V.R, et al, Goal Question Metric Approach, John Wiley&Sons,1994.
42. Mary Shaw Writing Good Software Engineering Research Papers.- Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, 2003, pp. 726-736.

Отримано: 25.05.2022

Про автора:

Сидоров Микола Олександрович,
доктор технічних наук,
професор.
Кількість наукових публікацій у українських виданнях – 140.
Кількість наукових публікацій у зарубіжних виданнях – 22.
http://orcid.org/0000-0002-3794-780X

Місце роботи автора:

Національний Технічний Університет України «Київський політехнічний інститут імені Ігоря Сікорського», факультет інформатики та обчислювальної техніки, кафедра інформатики та програмної інженерії, ІПІ, професор.
02000, Київ,
вул. Політехнічна, 41.
Моб. тел.: 067 7980361.
E-mail: nyksydorov@gmail/com