ДОСЛІДЖЕННЯ МОДЕЛЕЙ, МЕТОДІВ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ОЦІНКИ КВАЛІФІКАЦІЇ ПРОГРАМІСТА

УДК 004.81:159.953.52

DOI: https://doi.org/10.35546/2313-0687.2020.27.78-88

Катерина Полібіна,
аспірантка кафедри програмних засобів та технологій,
Херсонський національний технічний університет, Херсон, Україна,
E-mail: e.v.polibina@gmail.com, ORCID ID: 0000-0001-6727-1890

Анотація. На сьогодні в Україні попит на побудову кар'єри в IT зростає. Це обумовлено розвитком інформаційних технологій, і як слідство, розширенням ринку праці та зросту рівня заробітної платні в галузі IT. Асоціація «IT Ukraine» [1] розробила уніфікований перелік вимог та рекомендацій до спеціалістів junior-рівня. Перелік охоплює три найбільш популярні та затребувані IT-спеціалізації – розробку, тестування та автоматизоване тестування програмного забезпечення. Щоб почати роботу в IT, junior-розробник повинен володіти навичками програмування HTML/CSS на середньому рівні, JavaScript і .NET на початковому рівні [2].

Метою статті є дослідження можливостей використання концептуальної моделі структуризації та візуалізації знань з програмування для вивчення поза рамок формальної освіти та методу оцінки кваліфікації програміста-аматора за допомогою матриць Hexlet.

Методи дослідження. Концептуальна структуризація вирішує проблему побудови цілісного уявлення (концептуального каркаса) про галузь IT знань поза формальної освіти. Матриця Hexlet структурує компетенції програміста та тестує професійні компетенції. Аналіз статистичних даних використання сучасних IT технології та інструментів дозволяє структурувати дослідження в галузі популярності мов та інструментальних засобів програмування.

Основні результати дослідження. Концептуальна модель пропонує порядок вивчення предметів для самостійного опанування IT- спеціалізацією, що дає можливість зрозуміти, які дисципліни потрібні більше, які менше для роботи в певній галузі IT. Оптимізована матриця компетенцій дає розуміння набутої кваліфікації в програмуванні поза засвоєння фундаментальних основ.

Наукова новизна. Неформальне професійне навчання та підтвердження його результатів є новелою для України. Працюючі в IT-індустрії фахівці приділяють дуже багато уваги своєму професійному та особистісному розвитку, тому для адекватної оцінки навичок тут потрібні кастомізовані – «підігнані» під потреби IT галузі – методи.

Практична значимість. Для практичного підтвердження отриманих результатів проведено експеримент по опитуванню та тестуванню студентів Новокаховського приладобудівного фахового коледжу, які навчаються на економічних спеціальностях та самостійно набувають професійних компетенції в галузі IT. Результати експерименту підтвердили практичну цінність запропонованих моделей, які можуть бути використані для підтримки прийняття рішень по самостійному вивченню інформаційних технологій та оцінки набутої кваліфікації.

Ключові слова: *програміст, програмування, оцінка кваліфікації, самоосвіта.*

Постановка проблеми. Професія розробника програмного забезпечення (ПЗ) сьогодні є однією з найбільш затребуваних, високооплачуваних та з високою конкурентоспроможністю. Але щоб стати класним фахівцем, необхідно постійно самовдосконалюватися і вчитися. До вершин кар'єри в програмуванні можна прийти по-різному. Хтось розбирається в усьому самостійно, це, так звана самоосвіта, за допомогою книг, підручників, курсів, інтернет-ресурсів тощо. Інший варіант – піти в фаховий учбовий заклад і отримати диплом в області інформаційних технологій. Але головна проблема вищої освіти, складається в тому, що учбовий заклад не встигає за швидким розвитком інформаційних технологій. Програма навчання застаріває практично відразу після свого виходу в світ, так що студентам доводиться вчитися самостійно, щоб оставатися в тренді професії. Тому сучасні український програмісти-аматори не поспішають навчатися у навчальних закладах, всі прагнуть почати практикувати та заробляти якомога раніше. На жаль, частіше вивчається тільки один напрямок (наприклад PHP + Mysql). Та на цьому все і закінчується. В результаті ринок праці IT получає величезну кількість програмістів, які не знають базових речей. Звідси з'являються проблеми з якістю коду, з ефективністю алгоритмів, з низькою конкурентоспроможністю фахівців та створених ними ПЗ на галузевих ринках України та світу.

Аналіз останніх досліджень і публікацій. Тема самоосвіти в IT галузі, в сенсі неформального навчання та підтвердження його результатів, для України є маловивченою. Але ж, є де яки дослідження присвячені цьому питанню. Наприклад, викладач Одеського технічного коледжу Іванова Л.В. сумісно з колегою в своїй статті з власних наукових доробок [3] зазначає, що «…щомісяця українські IT-компанії оприлюднюють близько трьох тисяч нових вакансій. Нажаль, наразі все частіше мова йде про невідповідність існуючої системи підготовки фахівців реальним вимогам ринку та, зокрема, і у галузі інформаційних технологій». Російські колеги також опікуються питаннями підтвердження кваліфікації програміста, набутою через самоосвіту. Так, на приклад, в циклі своїх статей [4, 5], де в якості емпіричної бази дослідження виступають дані глибинних інтерв'ю з представниками IT-сфери, російська дослідниця Л.В. Земнухова робить висновки, що кваліфікаційний ресурс програміста розкривається через наявність і отримання постійного практичного досвіду. Що через швидкий розвиток професії виникають складнощі з її формалізацією, що призводить до множинної класифікацій працівників різних галузей IT. Крім того, це питання знаходиться в сфері уваги і зарубіжних науковців. В статті канадського дослідника Тімоті Літбриджа (Timothy C. Lethbridge) [6] розглядається необхідність надбання формальної освіти з точки зору навичок програмістів, які необхідні роботодавцю. Таки чином, сам факт існування терміну «програміст-аматор» [7] говорить про, що це питання вимагає ретельного дослідження.

Щодо моделей оцінки якості кваліфікації програмістів увагу привертає наукова праця білоруських вчених А.А. Прихожина та А.М. Ждановського. Науковці розглядають проблему формування і оптимізації команд програмістів з урахуванням кваліфікації і рівня володіння технологіями і інструментами програмування[8].

Мета дослідження. Мета дослідження полягає в вивченні та в доведенні доцільності використання концептуальної моделі структуризації та візуалізації знань з програмування для вивчення поза рамок формальної освіти та оптимізації методу оцінки кваліфікації програміста-аматора за допомогою матриць Hexlet.

Виклад матеріалу дослідження. Американська всесвітня соціальна платформа HackerRank з 1,5 млн користувачів, яка пропонує завдання різної складності з програмування, кожний рік проводить опитування [9]. Мета опитування визначити, наскільки важлива самоосвіта в галузі IT, а також дізнатися, які навички майбутні програмістаматори планують розвивати. На думку приблизно 50% респондентів, крім навчання в вузах, вони отримували необхідну інформацію самостійно. Більше 30% опитаних відмітили, що професійні навички отримали в процесі самонавчання, а не в навчальному закладі.

Крім того, HackerRank проаналізував вимоги роботодавців к рівню IT спеціалістів [10]. Як виявилося, найчастіше роботодавці (48% опитаних) беруть на роботу розробників, які знають JavaScript і суміжні фреймворки. Цей попит пов'язаний з активним переходом галузі на динамічні веб-сторінки, де 95% всіх додатків написані на JavaScript. Але ж лише 42% початківців-розробників знають цю мову. Найсильніше дефіцит таких кадрів в Україні і Канаді. У США і Великобританії, навпаки, фахівців з JavaScript цілком достатньо. Проблема в тому, що в багатьох освітніх закладах цю мову

програмування не включають в курс навчання. Ruby, Python і JavaScript – найпопулярніші мови. Всі три мови цікавлять переважна і більшість роботодавців. Початківцям-розробникам треба вивчати їх в обов'язковому порядку. Четвертою, за популярністю, залишається мова С, оскільки низькорівневі мови програмування допомагають у вивченні нових. Також, фреймворки AngularJS, React, Node.js і Spring – найпопулярніші серед роботодавців. Їх знання вимагає переважна більшість компаній. Дефіцит фахівців дуже помітний, оскільки саме ці фреймворки слабо вивчаються в освітніх закладах. Основи, звичайно, даються, але практики мінімум і студентам потрібно освоювати ці фреймворки самостійно.

Програмування – це повноцінна галузь знань, яка вимагає фундаментальної підготовки. Так, написати та підняти більшість сайтів можна прочитавши пару книг по PHP і HTML. Але «умовний» Google не створиш, не знаючи основ. Можливості для самоосвіти в комп'ютерних науках зараз величезні. Єдине, чого не вистачає, – це системності у підготовці. Як розібратися, що і в якій послідовності вивчати?

Для початку потрібно з'ясувати, яким програмістом хоче стати початківець або світчер. На IT-ринку затребувані як висококваліфіковані дорогі фахівці, так і «monkey-кодери». Пакет знань і досвіду перших і других відрізняється в значній мірі. Можна описати приблизний максимум знань, які так чи інакше відносяться до програмування. Звичайно, знати все неможливо – якесь питання потрібно знати глибоко, а в інших досить поверхневого оглядового розуміння. По-цьому в залежності від спеціалізації деякі дисципліни більш актуальні, деякі менш, але загальні базові знання необхідні для всіх з них, для будь-якого інженера-програміста.

Таким чином, нижче запропоновано концептуальну модель міждисциплінарних зв'язків при вивченні програмування. Очевидно, що одні дисципліни активно використовують знання інших дисциплін, або виростають з них. Відповідно для повного розуміння «верхнього» предмета, необхідний якийсь рівень розуміння «нижнього». Модель складається з дисциплін і розбита на рівні. Найнижчий рівень – «Загальна база», власне, відношення до комп'ютерних наук не має, але ж показує, на чому базуються вивчення дисциплін з програмування. Загалом, між наведеними дисциплінами існують 2 види зв'язків: використання (звичайна стрілка) і розширення (товста стрілка). Використання дисципліни має на увазі необхідність фрагментарних знань іншого предмета, а розширення – необхідність повних знань з дисципліни (рис.1).

Рівень «Спеціальна база знань» з CS (англ.computer science) – це стартовий майданчик для будь-якого програміста по чотирьох напрямках:

– арифметичні основи ЕОМ (системи числення і операції з числами, логічні операції); фізичні основи ЕОМ (напівпровідники, транзистори, логічні елементи, схеми, інтегральні мікросхеми);

– теорія алгоритмів (алгоритми і структури даних; складність, ефективність, способи представлення інформації в пам'яті);

– мови програмування (завдання і поняття МП, рівні, типи мов, абстракція, рівні абстракції, трансляція/компіляція, шаблони, принципи, парадигми – огляд).

Спеціальна база пропонує фундаментальні теоретичні знання, на яких будуються дисципліни вищих рівнів. Для середнього програміста необхідні оглядові знання з усіх предметів спеціальної бази. Для деякої спеціалізації потрібно поглиблене розуміння теорії алгоритмів (перш за все, розробникам різного роду бібліотек).

Рівнем вище розташовуються дисципліни, які є базовими саме в програмуванні – Основи програмування. У нього входять:

– архітектура ЕОМ (процесори, мікроархітектура, пам'ять, шини, введення/висновок);

– обробка інформації (теорія інформації, статистика, моделі, пошук даних, лінгвістичні аспекти, обробка інформації засобами електронних таблиць);

– основи С/С ++ (базові властивості мови, синтаксис, покажчики, введення/висновок, масиви, основи STL).

Рівень 1 – перший прикладний рівень, який дозволяє почати роботу по розробці ПЗ, оволодівши цим рівнем. Він включає 5 дисциплін:

– основи ASM (розвиток архітектури ЕОМ в напрямку програмування, написання найпростіших драйверів і алгоритмів, асемблерні вставки в С/С++);

– С/С ++ (ООП, розробка прикладних програм, бібліотеки, WinAPI, паралельне програмування);

– операційні системи (архітектура ОС, процеси, міжпроцесна взаємодія, потоки, планування, роботи з пам'яттю і периферією, POSIX-системи);

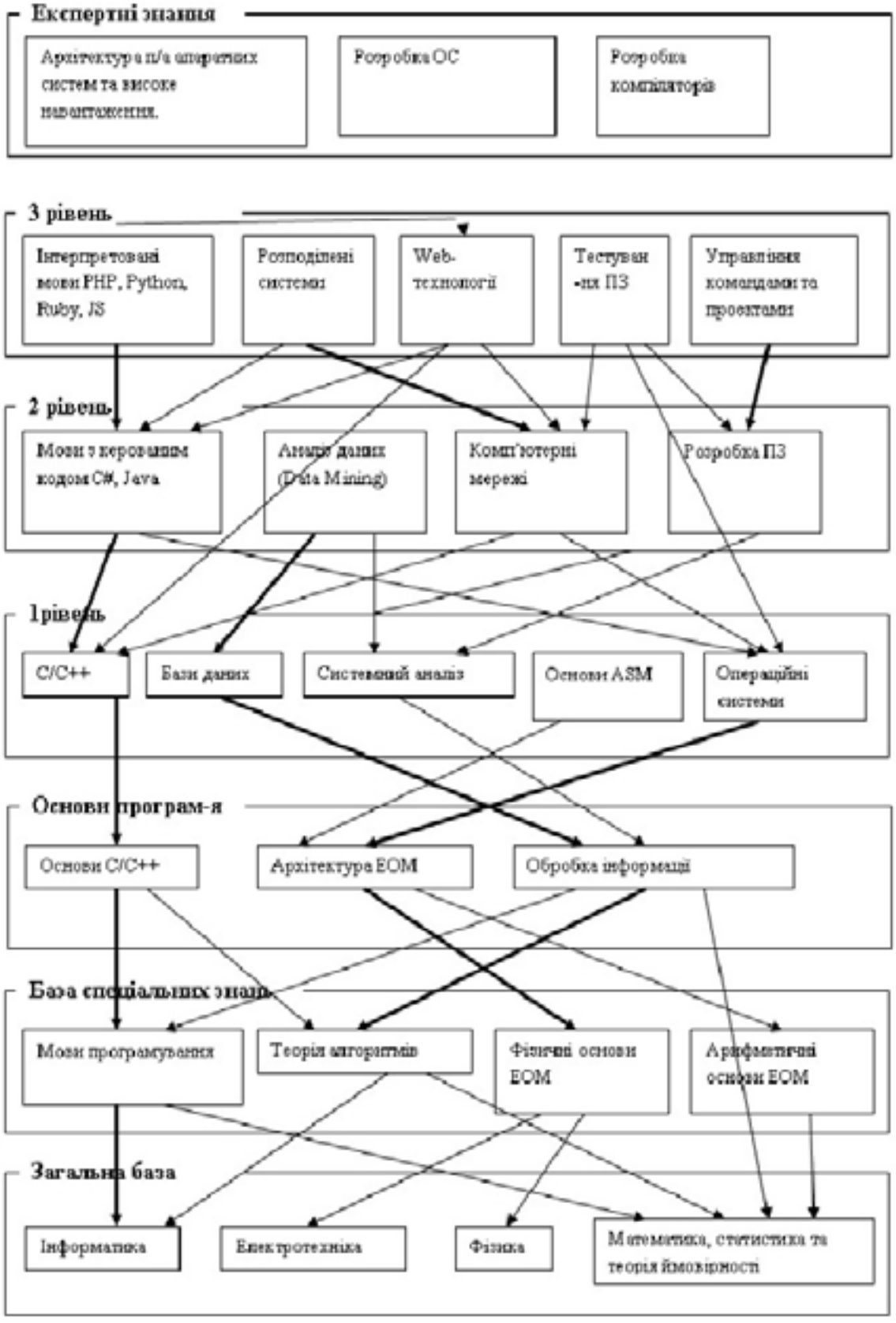

Рис. 1 – Концептуальна модель міждисциплінарних зв'язків при вивченні програмування

– системний аналіз (предметна область, бізнес-процеси, потоки, діаграми, принципи і теорія системного аналізу);

– бази даних (теорія множин, види СУБД, реляційні СУБД, моделі даних, SQL, конкретні БД).

Рівень 2 – розвиває попередній і включає:

– розробку ПО (життєвий цикл ПО, етапи розробки, основи ведення програмних проектів, інструменти);

– аналіз даних (Data Mining, OLAP, машинне навчання, нейронні мережі);

– комп'ютерні мережі (за рівнями стеків TCP/IP і, або, ISO/OSI, протоколи, мережеве програмування на C/C++. Комп'ютерні мережі потрапили в цей рівень з тієї причини, що для їх вивчення бажано попередньо освоїти операційні системи, а цей предмет ближче до першого рівня);

– мови програмування з керованим кодом (керований код, віртуальні машини, збирачі сміття, unit тестування, практика на C # або Java).

Рівень 3 – останній рівень для середнього програміста. Він самий об'ємний і включає тільки ті дисципліни, які безпосередньо пов'язані з розробкою ПЗ. Всього їх 6:

– розробка UI і usability (принципи побудови інтерфейсів користувача);

– управління командами та проектами (методології розробки та інші питання управління);

– тестування ПО (оглядово: види тестування, інструменти);

– веб-технології (HTTP-протокол, веб-сервер, CGI, кешування і проксінг, клієнтське програмування);

– розподілені системи (архітектури розподілених систем, протоколи мережевої взаємодії компонентів, інструменти, принципи, підходи до побудови розподілених систем, відмовостійкість, великі дані, високі навантаження);

– інтерпретовані мови програмування (особливості, основи по двом-трьом мовам, практика по одній-двома мовам: JS, PHP, Python, Ruby).

Все, що перераховано вище відноситься до експертних знань. Цей рівень можна розширювати необмежено, додаючи в нього суміжні з розробкою дисципліни і найбільш складні аспекти розробки ПЗ. В схемі наведено 3 приклади – розробка компіляторів, розробка операційних систем і побудова архітектури великих програмно-апаратних систем, або архітектури, розрахованих на особливо високі навантаження. Залежності до нижніх рівнів природно існують, але можна сказати, що всі наведені дисципліни нижчих рівнів так чи інакше зв'язані з експертним рівнем, тобто експертний рівень вимагає найширших знань і хорошого досвіду.

Таким чином, наведена модель пропонує порядок вивчення предметів, дає можливість зрозуміти, які дисципліни потрібні більше, які менше для роботи в певній спеціалізації (просто вибрати основний предмет спеціалізації і дивитися по зв'язках і віддаленості до інших). Дає розуміння, як самостійно вивчати комп'ютерні науки, якщо починати не з фундаментальних основ, а з прикладних знань (наприклад, PHP), для цього можна рухатися по зв'язкам в сторони і вниз.

Природно, що в разі самонавчання доведеться самостійно шукати джерела інформації та багато читати, щоб заповнити ті прогалини, яким зазвичай приділяють увагу на студентській лаві. У новачка-одинаки такої можливості часто немає, тому доведеться звертатися до книг. Тут можна порекомендувати в першу чергу прочитати книги з дискретної математики, з шаблонів проектування і проектування алгоритмів.

Та й книги – це хороше джерело інформації, однак люди – все ж істоти соціальні, тому ми добре засвоюємо інформацію, отриману від іншої людини. При самостійному навчанні ментора доводиться шукати самостійно. Таке спілкування важливо не тільки для того, щоб отримати нові знання, а й щоб не закинути навчання в цілому. Підтримку інших програмістів можна отримати на різних Інтернет- спільнотах розробників ПЗ, а також на інших ресурсах і форумах, на яких у новачка є можливість знайти відповідь на питання і висловити свою думку. Більше зусиль доведеться вкладати і в практику. Тут порадимо відразу починати роботу над осмисленням проектом, це позитивно позначиться на мотивації, можливість бачити, як розвивається власна програма, допоможе також самонавчання в ігровому форматі.

Отже, отримати необхідні для програміста знання і навички можливо і самостійно, але доведеться мати серйозну мотивацію, крім того, необхідно буде самостійно формувати навчальний план і балансувати між суто практичними навичками, які допоможуть прямо зараз, і фундаментальними знаннями, корисними при навчанні в цілому.

Основна ознака неформальної освіти – відсутність єдиних, в тій чи іншій мірі стандартизованих вимог до результатів навчальної діяльності.

Модель компетенцій – повний набір характеристик (професійних та особистісних), що дозволяє людині успішно виконувати відповідні функції і домагатися необхідних результатів. Ефективна модель має просту структуру, вона зрозуміла і легка для розуміння. Існує величезна кількість матриць оцінювання компетенцій програміста. Розглянемо саму популярну, яку пропонує База знань Hexlet [11]. Чотири рівня компетентності програміста, які умовно розкладені по 15 умовним «полицям», ось тільки декілька з них:

– знання алгоритмів;
– вміння організувати контроль версій;
– досвід проектування складних систем;
– читабельність коду і так далі [12].

Матриця замає багато місяці, з її повною версією можна ознайомитися на офіційному сайті Hexlet. Наведу матрицю компетенцій програміста для трьох рівнів junior, middle, senior, якій було оптимізовано для використання в статті (переклад с англійської автора). Кожен пункт це те, що необхідно знати. Кожен наступний стовпець включає всі знання з попереднього. Кожен наступний стовпець без знання попереднього, це ілюзія знання.

Таблиця 1 – Матрица компетентности программиста (Hexlet версія)

Компетенції/ Рівні	**Junior (початковий)**	**Middle (середній)**	**Senior (високій)**
Мови	Імперативна (php/js/ruby/python, java/c#/c/kotlin)	Lisp (clojure/racket)	Haskell, erlang, prolog
Асинхронне програмування	Промиси	Сопрограмми, Async/Await,	Модель акторів,999канали
До бази з коду	Робота з базою через драйвери	ORM ActiveRecord/Repository/ QueryBuilder	Data Mapper
Web	http, html	Мікрофреймворки, роутинг, шаблонизація	Фреймворки
Web- сервіси		Моделі роботи (process/ thread/event loop	Cgi, fastcgi
Експлуатація	Автоматизація ((ansible)	Міграції, Моніторинг, vagrant	Незмінна інфраструктура, контейнерна віртуалізація (docker)

До речі, подібні матриці іноді використовуються HR-фахівцями і teamleader для оцінки компетенцій співбесіді.

Таким чином, кожний рівень з таблиці практично відповідає рівням, наведеним в концептуальній моделі міждисциплінарних зв'язків при вивченні програмування самостійно.

Аналіз результатів досліджень компанії RedMonk [13] про популярність мов і інструментальні засоби програмування і результатів досліджень організації IEEE Spectrum [14] про рейтинг мов дозволив розробити таблицю, що описує 14 основних технологій і інструментів. Для кожної технології вказано рейтинг, який показує значимість і широту застосування технології.

За призначенням всі технологій ділиться на 5 підгруп:

1. Системи контролю версій і управління проектами включають Git, Jira з рейтингом 0.3 кожна.
2. До середовищ розробки відносяться Visual Studio і Eclipse, рейтинг обох 0.6.
3. Системи управління базами даних представлені Oracle SQL (рейтинг 0.5) і Microsoft SQL Server (рейтинг 0.6).
4. Мови програмування Java, C #, Visual Basic, C ++, Java Script і XSL з рейтингом 1.0, 0.9, 0.7, 0.9, 0.8 і 0.6 відповідно.
5. Операційні системи представлені Windows і Linux з рейтингом 0.6 і 0.5 відповідно.

Таблиця 2 – Ключові технології і інструменти програмування

Назва технології	Код	Рейтинг IEEE Spectrum	Рівень за концепт. мод.
Git 0.3	VGT	0.3	3
Jira	VJR	0.3	3
Visual Studio	DVS	0.6	2
Eclipse	DEC	0.6	3
Oracle SQL	OBM	0.5	3
Microsoft SQL Server	DBM	0.6	2
Java	LJ	1.0	3
C#	LC#	0.9	2
Visual Basic	LVB	0.7	1
C++	LCP	0.9	1
Java script	LJS	0.8	3
XSL	LXS	0.6	2
Windows	OSW	0.6	1
Linux	OSL	0.5	1

Для оцінки свого рівня володіння технологіями і інструментами програміcту-самоучці можна використати простий метод само опитування, тобто заповнити якусь форму, в якій вказати рівень володіння кожної з технологій. Рівень визначається за п'ятибальною шкалою: 0 – відсутність знання; 1 – мінімальне знання; 2 – проміжні навички; 3 – розширений досвід володіння технологією; 4 – знання і досвід експерта. Експертом вважається програміст, який володіє теоретичними експертними знаннями, розробив не менше двох крупних проектів і пропрацював не менше двох років з даною технологією. Програміст має розширені навички, якщо виконуються два критерії з трьох, і володіє проміжними навичками, якщо виконується один критерій.

В рамках експерименту було запропоновано студентам Новокаховського фахового приладобудівного технікуму, які навчаються на економічних спеціальностях та опановують програмування самостійно, використовуючи запропоновану модель, заповнити матрицю компетенцій.

Результати опитування 5 осіб представлені на рис. 2, де рядки відповідають технологіями з таблиці, а стовпці студентам, які вивчали програмування самостійно. В комірках таблиці стоїть цифра, що позначає самооцінювання студентом ступеню володіння компетенцією після самонавчання за п'ятибальною системою. Для візуалізації результату, вищий бал зі знання технологій виділено червоним кольором. Середній – зеленим, мінімальний – синій та жовтий. Дисципліну не вивчав – білий колір.

Студенти	Студент 1	Студент 2	Студент 3	Студент 4	Студент 5
Jira	0	0	0	0	0
Visual Studio	0	2	3	3	2
Microsoft SQL Server	0	0	3	4	3
Java	1	4	4	2	2
C#	3	3	0	3	3
Visual Basic	0	2	0	3	2
C++	0	4	0	1	1
Java script	4	1	4	1	1
XSL	2	1	2	2	1
Windows	3	4	3	3	4
Linux	0	0	2	0	0

Рис. 2 – Результати опитування програмістів-аматорів

Діаграма компетенцій – це пелюсткова діаграма, яка візуалізує чисельні показники компетенцій.

І це тільки найпростіші застосування матриці і діаграми компетенцій. Аналізуючи, можна побачити, що кожен зі студентів зробив вибір дисципліни виходячи з популярності інформаційної технології. Також вивчаємо мова або ПЗ, яку було вибрано для самостійного вивчення, не залежало від наявності фундаментальних знань у програміста-аматора. Можливо, що студент 3 може претендувати на позицію junior IT компанії,

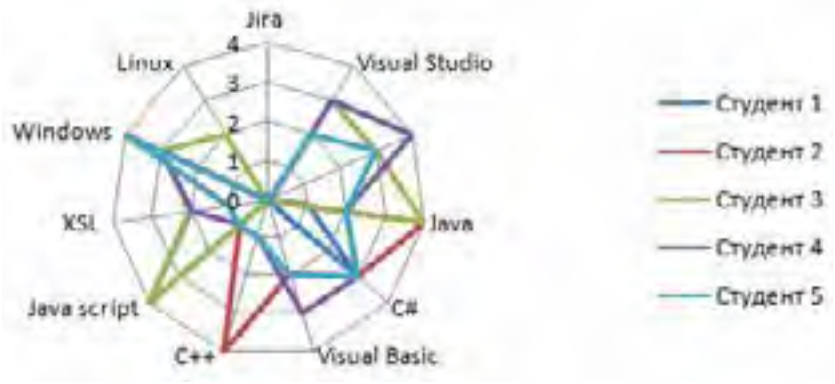

Рис. 3 – Пелюсткова діаграма компетенцій

не маючи диплома програміста. Також діаграма буде в нагоді, якщо студенти будуть об'єднані в команду для роботи над IT проектом

Висновки. Таким чином, стає зрозуміло, що диплом або ступінь не повинні бути основним фактором оцінки навичок спеціалістів, які намагаються построїти кар'єру в галузі IT. Основна увага при прийомі на роботу або участі молодого розробника в створенні ПЗ повинна приділятися індивідуальними проектами, його портфоліо, оцінці навичок.

Запропонована концептуальна модель структуризації знань вирішує проблему побудови цілісного уявлення (концептуального каркаса) про галузь IT знань поза формальної освіти. Оптимізована матриця Hexlet структурує компетенції програміста та тестує професійні компетенції і дає розуміння набутої кваліфікації в програмуванні поза засвоєння фундаментальних основ.

При цьому, незважаючи на те, що сучасні уявлення про освіту змінюються, багато роботодавців і фахівці (в тому числі в IT-сфері) все ще дивляться на диплом в тій чи іншій мірі. В IT-компаніях (зокрема, в Google) простіше потрапити на співбесіду з дипломом про вищу освіту в області комп'ютерних наук. Тому, є необхідність проаналізувати другий шлях здобуття навичок програмування, а саме – навчання в фахових освітніх установах. Але, як було сказано вище, головна проблема освіти в підготовці IT-спеціалістів в тому, що учбовий заклад не встигає за швидким розвитком технологій. Програма навчання застаріває практично відразу після свого виходу в світ. І запропонованим рішенням буде застосування нових освітніх технологій в професійній освіті. Це допоможе підняти рівень якості освіти IT, рівень цінності освіти IT, набуття практичних навичок майбутніми програмістами та досвіду роботи в команді, набуття зв'язків в галузі майбутньої праці.

СПИСОК ЛІТЕРАТУРИ:

1. IT Ukraine Association. Офіційний сайт об'єднання компаній-розробників програмного забезпечення в Україні. URL: https://itukraine.org.ua (дата звернення 15.01.2021).
2. Потрапити в IT: бізнес представив перелік ключових вимог до джуніорів. *Media та публікації*. 2018. URL: https://itukraine.org.ua/potrapiti-v-it-provn-kompan-rinku-sformuvali-dinij-perelk- klyuchovix-vimog-do-molodix-speczstv.html (дата звернення 25.12.2020).
3. Іванова Л.В., Скорнякова О.В. «Softskills», як важлива складова конкурентоспроможності фахівця з інформаційних технологій. *Young Scientist*. Херсон, 2018. № 12. С. 83–87.
4. Земнухова Л.В. «Айтишники» чаще любят свою работу: к обсуждению результатов исследования. *Петербургская социология сегодня*. 2013. № 4. С. 88–115.
5. Земнухова Л.В. Информационные технологии как профессиональная среда. *Социологический журнал*. 2013, № 4. С. 50–58.
6. Timothy C. Lethbridge. The Relevance of Software Education: A Survey and Some Recommendations. *Article in Annals of Software Engineering*. 2000. № 5. P. 3-19.
7. Столяр С.Е. Интернет-школа программирования. *Компьютерные инструменты в образовании*. 2015. № 1. С. 55–62.
8. Пригожий А.А., Ждановский А.М. Метод оценки квалификации и оптимизация состава профессиональных групп программистов. *Системный анализ и прикладная математика*. 2018. № 2. С. 4–11.

9. HackerRank: Official website the platform to identify and hire developers. URL: https://www.hackerrank.com (дата звернення 15.01.2021).
10. Рейтинг HackerRank: самые лучшие программисты — в Китае, России и Польше. *Статистика в IT*. URL: https://habr.com (дата звернення 15.01.2021).
11. Hexlet: Сайт по обучению программированию. URL: https://ru.hexlet.io (дата звернення 24.01.2021).
12. Computer Science: Programmer Competency Matrix. URL: https://sijinjoseph.com/programmer-competency-matrix (дата звернення 15.01.2021).
13. Red Monk: Official website of the Red Monk analytical company. Retrieved from http://redmonk.com (дата звернення 15.12.2020).
14. Cass S. Top Programming Languages (IEEE Spectrum). URL: http://spectrum.ieee.org (дата звернення 10.01.2021).

RESEARCH OF MODELS, METHODS AND INFORMATION TECHNOLOGIES FOR ASSESSING THE PROGRAMMER'S QUALIFICATION

Ekaterina Polibina,
PhD student of the Department of Programming Tools and Technologies,
Kherson National Technical University, Kherson, Ukraine,
e-mail: e.v.polibina@gmail.com, ORCID ID: 0000-0001-6727-1890

Abstract. Today in Ukraine the demand for building a career in IT is growing. This is due to the development of information technology, and as a result, the expansion of the labor market and the growth of wages in the IT field. Association IT Ukraine [1] has developed a unified list of requirements and recommendations for junior-level specialists. The list includes the three most popular and demanded IT specializations – development, testing and automated software testing. To get started in IT, a junior developer must have intermediate level HTML / CSS programming skills, beginner level JavaScript and NET skills [2].

The purpose of the article is to study the possibilities of using the conceptual model of structuring and visualizing knowledge in Programming for studying outside the framework of formal education and the method of assessing the qualifications of an amateur programmer using Hexlet matrices.

Research methods. Conceptual structuring can solve the problem of building a holistic view (conceptual framework) in the field of IT knowledge outside of formal education. The Hexlet matrix structures the programmer's competencies and tests professional competencies. Analysis of statistical data on the use of modern IT technologies and tools helps to structure research in the field of the popularity of languages and programming tools.

The main results of the study. The proposed model identifies the order of studying subjects for independent mastering skills in IT, which makes it possible to understand which disciplines are more needed and which ones are less important for work in a certain area of IT. The optimized matrix provides insight into acquired programming skills beyond the fundamentals.

Scientific novelty. Informal vocational training and recognition of its results is a novelty for Ukraine. People working in the IT industry pay a lot of attention to their professional and personal development, therefore, for an adequate assessment of skills, they need customized methods – "tailored" to the needs of the IT industry.

Practical significance. For practical testing of the results obtained, an experiment was carried out on the basis of the Nova Kakhovka Instrument-Making Vocational College. The students who study Economics and independently acquire professional competencies in the field of IT were interviewed and assessed according to the proposed model. The results of the experiment testify the practical value of the proposed models, which can be used to support decision-making on the independent study of information technologies and assess the acquired qualifications.

Key words: *programmer, programming, qualification assessment, self-education*

ИССЛЕДОВАНИЕ МОДЕЛЕЙ, МЕТОДОВ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ОЦЕНКИ КВАЛИФИКАЦИИ ПРОГРАММИСТА

Екатерина Полибина,
аспирантка кафедры программных средств и технологий,
Херсонский национальный технический университет, Херсон, Украина,
e-mail: e.v.polibina@gmail.com, ORCID ID: 0000-0001-6727-1890

Аннотация. На сегодня в Украине спрос на построение карьеры в IT растет. Это обусловлено развитием информационных технологий, и как следствие, расширением рынка труда и роста уровня заработной платы в области IT. Ассоциация «IT Ukraine» [1] разработала унифицированный перечень требований и рекомендаций к специалистам junior-уровня. Перечень включает три наиболее популярные и востребованные IT-специализации – разработку, тестирование и автоматизированное тестирование программного обеспечения. Чтобы начать работу в IT, junior-разработчик должен обладать навыками программирования HTML / CSS на среднем уровне, JavaScript и NET на начальном уровне [2].

Целью статьи является исследование возможностей использования концептуальной модели структуризации и визуализации знаний по программированию для изучения вне рамок формального образования и метода оценки квалификации программиста-любителя с помощью матриц Hexlet.

Методы исследования. Концептуальная структуризация решает проблему построения целостного представления (концептуального каркаса) в области IT знаний вне формального образования. Матрица Hexlet структурирует компетенции программиста и тестирует профессиональные компетенции. Анализ статистических данных использования современных IT технологий и инструментов позволяет структурировать исследования в области популярности языков и инструментальных средств программирования.

Основные результаты исследования. Предложенная модель предлагает порядок изучения предметов для самостоятельного освоения IT специализации, что позволяет понять, какие дисциплины нужнее, а какие – менее значимы для работы в определенной области IT. Оптимизированная матрица дает понимание приобретенной квалификации в программировании вне усвоения фундаментальных основ.

Научная новизна. Неформальное профессиональное обучение и подтверждение его результатов является новеллой для Украины. Работающие в IT-индустрии специалисты уделяют очень много внимания своему профессиональному и личностному развитию, поэтому для адекватной оценки навыков здесь нужны кастомизированные – «подогнанные» под нужды IT отрасли методы.

Практическая значимость. Для практического подтверждения полученных результатов, проведен эксперимент на базе Новокаховского приборостроительного профессионального колледжа. Были опрошены и протестированы студенты, которые учатся на экономических специальностях и самостоятельно приобретают профессиональные компетенции в области IT по предложенной модели. Результаты эксперимента подтвердили практическую ценность предложенных моделей, которые могут быть использованы для поддержки принятия решений по самостоятельному изучению информационных технологий и оценки приобретенной квалификации.

Ключевые слова: *программист, программирование, оценка квалификации, самообразование.*

REFERENCES:

1. IT Ukraine Association (2021). *Official website software development companies*. Retrieved from https://itukraine.org.ua (in Ukr.)
2. IT Ukraine Association (2018). *Get into IT: The business has introduced a list of key requirements for juniors*. Retrieved from https://itukraine.org.ua/potrapiti-v-it-provn-kompan-rinku-sformuvali-dinij-perelk-klyuchovix-vimog-do-molodix-speczstv.html(in Ukr.)
3. Ivanova, L.V., Skornyakova, O.V. (2018). Softskills as an important component of the competitiveness of an information technology specialist. *Young Scientist*, 12, 83–87. (in Ukr.).

4. Zemnukhova, L.V. (2013). IT specialists are more likely to love their job: to discussing research results. *Peterburgskaya sotsiologiya segodnya (Petersburg Sociology Today)*, 4, 88–115. (in Russ.).
5. Zemnukhova, L.V. (2013). Information technology as a professional environment. *Sotsiologicheskiy zhurnal (Sociological Journal)*, 4, 50–58. (in Russ.).
6. Lethbridge, Timothy C. (2000). The Relevance of Software Education: A Survey and Some Recommendations. *Annals of Software Engineering*, 5, 3–19.
7. Stolyar, S.E. (2015). Internet School of Programming. Kompyuternyie instrumentyi v obrazovanii (*Computer Tools in Education*), 1, 55–62.
8. Prihozhy, A.A., Zhdanovsky, A.M. (2018). Method for assessing qualifications and optimization of professional groups of programmers. *Sistemnyiy analiz i prikladnaya matematika (Systems Analysis and Applied IT)*, 2, 4–11. (in Rus.).
9. HackerRank (2021). *Official website the platform to identify and hire developers*. Retrieved from https://www.hackerrank.com (in English).
10. Habr, Statistics in IT (2021). *HackerRank rating: the best programmers – in China, Russia and Poland*. Retrieved from https://habr.com (in Rus.).
11. Hexlet (2021). *Teaching programming*. Retrieved from https://ru.hexlet.io (in Russ.).
12. Computer Science (2021). *Programmer Competency Matrix*. Retrieved from https://sijinjoseph.com/programmer-competency-matrix (in English).
13. Red Monk (2021). *Official website of the Red Monk analytical company*. Retrieved from http://redmonk.com (in English).
14. Cass, S. IEEE Spectrum (2020). *The 2020 Top Programming Languages*. Retrieved from http://spectrum.ieee.org (in English).