УДК 004.43

Сапожник Д.О., аспірант
Державний університет «Житомирська політехніка»

Квантовий гейт Адамара, реалізація у мові програмування Q#

Квантові технології відкривають можливості для дослідження нових алгоритмів для квантового програмування які будуть вирішувати надскладні обчислення набагато швидше ніж класичні комп'ютери. Основою таких алгоритмів та обчислень є квантові вентилі (квантові логічні елементи), які, на відміну від класичних логічних вентилів, дозволяють працювати зі станами суперпозиції та квантового заплутування. Одним з ключових вентилей є вентиль Адамара, що дозволяє створювати суперпозицію квантових станів[1].

Гейт Адамара — це один із ключових квантових гейтів, який відіграє важливу роль у створенні стану суперпозиції кубіта. Щоб зрозуміти принцип його роботи, давайте спершу розберемося з концепцією суперпозиції[1].

```
namespace HGateSample {
    @EntryPoint() // Позначення точки входу
    Run | Histogram | Estimate | Debug | Circuit
    operation Main() : Result {
        use q = Qubit(); // Створення кубіта
        H(q); // Застосування гейта Адамара

        let result = M(q); // Вимірювання кубіта

        Reset(q); // Повернення кубіта у стан |0>
        return result; // Повернення результату вимірювання
    }
}
```

Рис. 1. Приклад застосування вентиля Адамара мовою програмування Q#

Дані з якими працює класичний комп'ютер мають вигляд бітів, які можуть бути або в стані 0, або в стані 1. У квантовому комп'ютері дані представлені у вигляді кубітів, які можуть бути в обох станах (0 і 1) одночасно. Цей одночасний стан називається суперпозицією[1].

Роботу вентиля Адамара можна описати наступними кроками:

1. Вентиль Адамара приймає кубіт, який знаходиться у відомому класичному стані — зазвичай це стан 0 або стан 1.
2. Застосування вентиля Адамара змінює стан кубіта на такий, у якому кубіт стає у суперпозицію між 0 та 1.
3. У стані суперпозиції кубіт має певну ймовірність бути виміряним як 0 та певну ймовірність бути виміряним як 1. Гейт Адамара створює однакову ймовірність того, що кубіт буде виміряний як 0 або як 1 (кожен по 50%)

Мова програмування Q#, розроблена корпорацією Microsoft у рамках квантового обчислювального пакета Quantum Development Kit (QDK), надає зручний інтерфейс для реалізації квантових алгоритмів[2].

Наведений приклад коду на рисунку 1 демонструє використання вентиля Адамара для переведення кубіта у стан суперпозиції. У наведеному коді використовуються наступні елементи:

- using - оператор, який виділяє кубіт для квантових обчислень.
- H - застосування гейта Адамара.
- M - вимірювання стану кубіта.
- Reset - повернення кубіта у початковий стан.
- return - повернення результату (зображений на рисунку 2)

.

```
Result: "Zero"
Finished shot 1 of 1

Q# simulation completed.
```

Рис. 2. Результат виконання коду

Гейт Адамара відіграє важливу роль у багатьох квантових алгоритмах, зокрема, у квантовому алгоритмі Шора та алгоритмі Гровера. Розуміння принципів роботи та реалізація цього гейта у мові програмування Q# є основою для подальших досліджень у галузі квантових обчислень.

Список використаних джерел

1. Вступ до квантового обчислення та квантової інформації. — ЛНУ імені Івана Франка, 2021 [Електронний ресурс]. – Режим доступу : http://ktf.lnu.edu.ua/books/Krokhmalskii-VKO.pdf
2. Документація Q# [Електронний ресурс]. – Режим доступу : https://microsoft.github.io/qsharp/

УДК 621.37:621.391

Ципоренко В.Г., к.т.н., доц.
Курдиш А.С., студент, гр. ІВ-20-1
Криворучко М.А., студент, гр. ІВ-20-1
Державний університет «Житомирська політехніка»

А сенсорна станційна система охорони залізничних вантажів

На сьогодні найбільш надійним та універсальним засобом доставки товарів та вантажів є залізничний транспорт. Важливими чинником якості перевезення вантажів є гарантованість їх доставки, цілісність та неушкодженість. Ці вимоги забезпечують відповідні заходи та системи по охороні, супроводу та збереженню вантажів.

Вантажі на залізниці найбільш уразливі в місцях стоянки потягів, місцях їх технологічного переформування, а також на ділянках шляху з уповільненим рухом. Особливостями умов перевезення вантажів на залізниці є великі відстані транспортування, можливість зміни складу та порядку з'єднання вагонів потягу.

Аналіз показав, що на сьогодні основними варіантами контролю та упередження несанкціонованого доступу до вантажів є наступні: використання пристроїв блокування несанкціонованого доступу, патрульна охорона вантажів на станціях, використання потягової охорони особовим складом в русі по маршруту, застосування автоматизованих систем дистанційного позиціювання та контролю пересування вантажів по маршруту на основі мережевих та супутникових технологій, використання радіомаркерів, засобів ідентифікації та пошуку вантажів.

При використанні патрульної охорони вирішують задачі аналітичного аналізу особливості маршруту перевезення і використовують різні види охорони: при русі потягу, на станціях сортування і транзиту. Основним недоліком патрульної охорони є висока ціна, складність організації охорони та недостатня надійність.

Основним недоліком сучасних супутникових систем дистанційного позиціювання та супроводу вантажів є недостатня швидкодія реагування на несанкціоновані доступи, а також складність оперативної взаємодії з підрозділами охорони.

Аналіз мережевого зв'язку загального користування показав, що його основною перевагою є широкі функціональні можливості, багаторівнева система безпеки, можливість автоматичного перезавантаження системи при виникнення збоїв.

Проте суттєвими недоліками застосування такого типу існуючих мереж є відсутність гарантії неперервного зв'язку, відносно низька надійність з'єднання та відсутність достатньої оперативності зв'язку.

На відміну від існуючих рішень, авторами запропоновано сенсорну станційнусистему охорони вантажів потягів на основі використання технологій цифрових вузькосмугових радіомереж. Показано, що сучасні технічні засоби дозволяють створювати автономні, недорогі, ефективні і гнучкі охоронні радіомережі вантажів на території станцій, що забезпечують функціонування в реальному масштабі часу, з високою надійністю та мінімальними витратами на обслуговування.

Джерелами даних такої системи доцільно використати групу датчиків безпеки об'єктів потягів, що обладнані відповідним радіомодемом. Обмін даними виконується з пристроєм керування (пультом), що працює з базовою станцією по відповідним протоколам.

Показано, що систему охорони доцільно будувати по одноранговій топології з використанням технології Mesh Grid. В цьому випадку просторово віддалені датчики можуть не мати прямого зв'язку з центральним вузлом збору даних. Інформація буде передаватися по радіорелейному ланцюжку через довільні доступні суміжні модулі, що розташовані ближче до базової станції.

Суттєвою перевагою використання автономної мережі є можливість високого рівня безпеки даних по перехопленню, несанкціонованому доступу та завадозахищеності, якість якої може перевищувати якість кабельних мереж зв'язку.

Запропоновано реалізувати систему охорони потягу як автономну мережу ISM-діапазону. Це забезпечує мінімізацію потужності споживання, збільшену дальність дії, підвищену завадозахищеність. Технічною базою для реалізації такої системи доцільно використати цифрові трансивери серії CCxxx фірми Chipcon.

Перевагою цього варіанту порівняно з відомими мережами архітектури WLAN є підвищена завадозахищеність, енергетична і спектральна ефективність з можливістю автономного батарейного живлення, невисока собівартість, простота протоколів обміну і програмного забезпечення, підвищена дальність дії, здатність підтримувати мережеву технологію Mesh Grid.

Виконані розрахунки та дослідження характеристик запропонованої сенсорної станційної системи охорони вантажів потягів підтвердили ефективність її застосування.

УДК 004.4

Волков О. Г., здобувач
Савіцький Р. С., аспірант, ст. викладач,
Державний університет «Житомирська політехніка»

Проєктування та аналіз вимог програмних вебсистем жеребкування турнірів

Організація та проведення спортивних турнірів є комплексною задачею та вимагає удосконалених підходів до організації. З метою спрощення та оптимізації процесу та забезпечення справедливого та ефективного розподілу учасників, розробляється веб-система жеребкування турнірів.

Водночас, організація спортивних змагань потребує удосконалення технологій, конкретніше, розвинення системи турнірів та збільшення числа учасників, а разом з тим, постає необхідність у зручних та ефективних інструментах для керування подіями. Розробка системи для проведення жеребкування турнірів стає ключовим етапом для успішного та справедливого проведення спортивних змагань. Головна мета проєкту полягає у створенні високоякісної та функціональної веб-системи, що відповідатиме потребам сучасної спортивної громадськості.

Враховуючи вищеперелічені фактори, було спроєктовано наступні функціональні вимоги, яким має відповідати система:

ефективне проведення жеребкування та розподіл учасників, заощаджуючи час та зусилля організаторів;

справедливе розподілення учасників за допомогою алгоритмів жеребкування, що допомагає уникнути будь-яких підозр в необ'єктивності;

зручна та прозаїчна реєстрація для участі в турнірі та отримання інформації про свої матчі та подальші етапи через зручний інтерфейс системи;

можливість вибору різних алгоритмів та параметрів жеребкування, що дозволяє адаптувати систему під конкретні потреби та особливості турніру;

забезпечення прозорості та доступності інформації про жеребкування та результати для всіх учасників турніру;

мінімізація витрат на організацію та проведення турнірів завдяки автоматизації процесу жеребкування та управління учасниками.

Звернімо увагу на те, що автоматизація турнірного управління має на меті покращити ефективність та зручність самого продукту. Вона дозволить не лише оптимізувати процеси, але й забезпечити високий рівень задоволеності серед учасників. Завдяки точно спланованим алгоритмам можна значно зменшити час на підготовку та проведення етапів, а також уникнути помилок, які можуть зародитись внаслідок неадаптованої автоматизації.

Саме тому варто зауважити, що важливість деяких компонентів, які потрібно автоматизувати, не може бути ігнорована:

- автоматичне формування розкладу матчів та етапів турніру на основі введених даних про учасників та параметрів турніру;
- автоматичне сповіщення учасників про результати жеребкування, їхні матчі та подальші інструкції через електронну пошту або повідомлення в додатку;
- визначення оптимального часу для проведення кожного етапу турніру та розподіл ресурсів (спортивних майданчиків, арбітрів тощо) з урахуванням розкладу матчів;
- можливість автоматичної корекції розкладу турніру в разі змін учасників, відмінених матчів або інших факторів.

З огляду на вимоги створення веб-системи для жеребкування в турнірах, розглянемо обрані технології та їх обґрунтування. На бекенді було обрано мову програмування C#. Вона є потужною та ефективною мовою, яка добре підходить для розробки веб-застосунків. Вона має велику спільноту розробників та підтримується Microsoft, а також дозволяє реалізувати складну логіку, включаючи алгоритми жеребкування та обробку даних.

Було б неправильним вказавши, що було обрано C#, не вказати про те, що було обрано фреймворк ASP.NET Core, який в свою чергу, є відкритим та кросплатформенним фреймворком для розробки веб-застосунків; він надає можливість побудови високопродуктивних та масштабованих додатків. Також важливим аспектом являється те, шо ASP.NET Core дозволяє реалізувати RESTful API для взаємодії з фронтендом.

Відповідно, на фронтенді було запропоновано використати React.js. React.js є популярною бібліотекою для створення інтерфейсу користувача, що дозволяє побудувати динамічний та реактивний інтерфейс. React має велику спільноту розробників та багато готових компонентів. Додатково, HTML використовується для структуризації контенту веб-сторінок та CSS дозволяє оформити контент, надаючи йому стиль та вигляд. Говорячи про контент сторінок, потрібно брати підходящу інформацію, аби заповнити саме їх. Звідси випливає, що потрібна база даних для ефективного розподілу бази даних. Враховуючи всі за та проти, було обрано SQL Server в ролі системи менеджменту реляційної бази даних. Він є надійною та потужною системою керування базами даних, що дозволяє зберігати дані про турніри, учасників, результати жеребкування та іншу інформацію. SQL Server підтримує транзакції, індексацію та запити для ефективного доступу до даних. Фіналізуючи, даний стек технологій дозволить лаконічно реалізувати функціонал системи жеребкування для турнірів, забезпечити високу продуктивність, безпеку та зручний користувацький досвід.

Список використаних джерел

1. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages / Adam Freeman, 2022 - 1267 с

УДК 004.4

Баламут Д.О.
Вакалюк Т.А., зав.каф.інженерії програмного забезпечення, д-р пед. наук, проф.
Державний університет «Житомирська політехніка»

Бібліотеки Tensorflow, Kares, Opencv, Numpy та Scipy у Python: можливості та застосування для аналізу розпізнавання емоцій обличчя

В останні десятиліття штучний інтелект та машинне навчання стали невід'ємною частиною нашого повсякденного життя. Одним з напрямків, де ці технології знаходять широке застосування, є аналіз розпізнавання емоцій на обличчі людини. Ця галузь знайшла своє застосування в різних сферах, від відеоігор та розваг до медицини та психології. Одним з ключових інструментів для реалізації завдань з розпізнавання емоцій є бібліотека TensorFlow, що надає широкі можливості для розробки та впровадження моделей штучного інтелекту.

Проаналізувавши різні аспекти використання TensorFlow для аналізу розпізнавання емоцій на обличчі, можна виявити ряд переваг та цікавих можливостей, які ця бібліотека пропонує. Далі в статті ми розглянемо деякі з них та проаналізуємо приклади застосування TensorFlow у цій конкретній галузі.

Python є однією з 10 найкращих мов програмування у світі сьогодні. Ця мова дуже популярна в спільноті учнів і працівників. Google, Facebook, Instagram зараз використовують Python для програмування своїх великих і малих проектів. Python має дуже простий, чистий синтаксис. Його набагато легше читати та писати порівняно з іншими мовами програмування, такими як C++, Java, C#. Оскільки Python є програмою з відкритим кодом, розробники можуть не лише використовувати програмне забезпечення та програми, написані на Python, а й змінювати вихідний код. Все, що не можна вирішити іншими мовами програмування, легко вирішується в Python. [1]

Глибоке навчання є частиною машинного навчання, яке базується на використанні складних нейронних мереж. Глибоке навчання використовує цю штучну нейронну мережу для аналізу даних, прогнозування та представлення ситуацій на основі аналізу та «досвіду» на основі вивчених шаблонів так само, як люди сприймають об’єкт або явище, з якими вони стикалися. Мова програмування Python відома своєю простотою, простим у вивченні та зрозумілим кодом, логічним і лаконічним синтаксисом, тоді як машинне навчання передбачає надзвичайно складні алгоритми та багатоетапні робочі процеси. Лаконічна та проста логіка Python відіграє важливу роль у економії часу та одночасному здобутті переваги пропозиції кращих рішень. [2]

Назва TensorFlow безпосередньо походить від його основної структури: Tensor. У TensorFlow усі обчислення включають тензори. Тензор — це вектор або матриця n вимірів, що представляють усі типи даних. Усі значення в тензорі містять ідентичний тип даних із відомою або частково відомою формою. Форма даних — це розмір матриці або масиву (Роман 2020). TensorFlow дозволяє розробникам будувати графіки та структури потоків даних, які визначають, як дані переміщуються через графік, беручи в якості вхідних даних багатовимірний масив, який дозволяє користувачам будувати діаграму операцій, які можна виконувати над цими вхідними даними, переходячи на один кінець. і надходять до іншого як виходи. [2]

TensorFlow — це наскрізна бібліотека з відкритим вихідним кодом, створена переважно для програм машинного навчання. Це символічна математична бібліотека, яка використовує розрізнене програмування та потоки даних для виконання різноманітних завдань, зосереджуючись на навчанні та виведенні глибоких нейронних мереж. Це дозволяє розробникам створювати програми для машинного навчання, використовуючи різні інструменти, бібліотеки та ресурси спільноти. TensorFlow був створений і розроблений командою дослідників Google (Google Brain) і офіційно ліцензований на роботу 9 листопада 2015 року з метою підтримки досліджень і застосування в ефективному виробництві. [2].

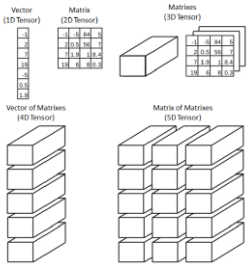


Рис. 1. Тензорна архітектура

Кожна програма TensorFlow сама є програмою Python і написана у високопродуктивних двійкових файлах C++. TensorFlow покладатиметься на мову Python, щоб надати програмістам усе вищезазначене, оскільки Python простий у вивченні та допомагає зрозуміти, як зробити абстракції високого рівня складними. [2] Keras був прийнятий TensorFlow як офіційний API високого рівня. Якщо його вбудовано в

TensorFlow, він робить доступними модулі для всіх обчислень нейронних мереж і, таким чином, може виконувати глибоке навчання дуже швидко. TensorFlow є дуже гнучким і головною перевагою є розподілене обчислення, тож розробники можуть бути гнучкими та контролювати свою програму, реалізовувати ідеї за короткий час, використовуючи Keras, тоді як обчислення включають тензори, обчислювальні графіки та сеанси, які можна налаштувати за допомогою Tensorflow Core API. [3]

Keras — це програма з відкритим вихідним кодом для нейронної мережі, написана мовою Python. Це бібліотека, розроблена в 2005 році Франсуа Шолле, дослідником Deep Learning. Keras — це API високого рівня, розроблений для Python, щоб полегшити впровадження нейронних мереж. Keras може працювати на бібліотеках і фреймворках, таких як Tensor-Flow і Theano. Вони є дуже потужними, але також заплутаними бібліотеками для створення нейронних мереж. З іншого боку, Keras дуже зручний для початківців, оскільки його мінімальна структура забезпечує чистий і простий спосіб генерувати моделі глибокого навчання на основі TensorFlow або Theano. [3] OpenCV надає понад 2500 класичних і сучасних алгоритмів, оптимізованих для машинного навчання та комп'ютерного зору. І оскільки вона була написана на оптимізованому C/C++, бібліотека може скористатися перевагами багатоядерної обробки. Основним об'єктом OpenCV є програми реального часу. [4] OpenCV — це абревіатура фрази «Open-Source Computer Vision Library» — бібліотека з відкритим кодом для машинного навчання та комп'ютерного зору. OpenCV створено для аналізу в реальному часі та машинного навчання, а також для обробки зображень і відео. Наразі набір інструментів OpenCV також додано з прискоренням графічного процесора в реальному часі. Проект OpenCV народився в 1999 році, створений Гері Бредскі з Intel. У 2000 році була випущена перша версія OpenCV. OpenCV випускається за ліцензією BSD (Berkeley Soft-ware Distribution), тому він абсолютно безкоштовний як для академічного, так і для комерційного використання. Він має інтерфейси C++, C, Python, Java і підтримує Windows, Linux, Mac OS, iOS і Android. OpenCV розроблено для ефективних обчислень і зосереджено на програмах реального часу.

SciPy розшифровується як Science Python і є безкоштовною науковою бібліотекою Python з відкритим вихідним кодом для математики, науки та інженерії. Бібліотека SciPy побудована на основі бібліотеки NumPy, забезпечуючи зручне та швидке маніпулювання N-вимірним масивом. SciPy включає підмодулі для лінійної алгебри, оптимізації, інтеграції та статистики. SciPy є базовою бібліотекою, яка створена на основі розширення NumPy. SciPy сумісний з об'єктом N-вимірного масиву NumPy. SciPy містить код для роботи функцій NumPy. SciPy і NumPy разом — найкращий вибір для наукових операцій. [5] NumPy означає Numerical Python. NumPy — це математична бібліотека, яка використовується для роботи з матрицями та масивами та зазвичай використовується для роботи з великими масивами та матричними даними. NumPy був створений у 2005 році Тревісом Оліфантом, і його швидкість обробки в рази вища, ніж використання звичайних масивів і списків Python. NumPy надає об'єкти та методи для роботи з багатовимірними масивами та операціями лінійної алгебри. У NumPy розмірність масиву називається осями, тоді як розмірність називається рангом. Основною бібліотекою в NumPy є об'єкти масиву. Масиви схожі на списки в Python, за умови, що кожен елемент у масиві повинен мати однаковий тип даних. Масиви можуть працювати з великими обсягами числових даних, як правило, із числами з плаваючою речовиною або цілими числами, і набагато ефективніші зі списками. Зазвичай у NumPy використовується клас Ndarray (N-вимірний масив).

У результаті дослідження з використанням Python та бібліотек TensorFlow, Keras, OpenCV, NumPy та SciPy для аналізу та розпізнавання емоцій обличчя було виявлено високий потенціал цих інструментів у сферах штучного інтелекту та комп'ютерного зору. Застосування нейронних мереж та комп'ютерного зору дозволяє ефективно визначати емоційний стан особи на основі зображення обличчя. Це може мати важливе значення у різних галузях, таких як психологія, медицина, безпека та розваги. Розширення можливостей цих технологій може сприяти подальшому розвитку систем інтелектуального аналізу та взаємодії з людьми, забезпечуючи нові можливості для покращення якості життя та розвитку різних сфер діяльності.

Список використаних джерел

1. LearnPython. Онлайн ресурс для навчання Python. Доступно за посиланням: https://learnpython.com/blog/. Останній доступ: 9 лютого 2023 року.
2. Ketkar, N., & Moolayil, J. (2021). Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch. New York City: Apress.
3. Chollet, F. (2018). Deep Learning with Python. New York: Manning Publications.
4. Bradski, G., & Kaehler, A. (2008). Learning OpenCV. California: O'Reilly Media.
5. The SciPy Community. (2013). SciPy Reference Guide. Доступно за посиланням: https://docs.scipy.org/doc/scipy-0.13.0/scipy-ref.pdf

УДК 621.37

Ципоренко В.В., к.т.н., доц.
Ципоренко В.Г., к.т.н., доц.
Денисюк М.С., бакалаврант, гр. ІВ-20-1
Державний університет «Житомирська політехніка»

Розроблення бездротової мережі відеоспостереження

Бездротові камери спостереження – камери, які передають відео та аудіо сигнал на бездротовий приймач через радіодіапазон. Багато бездротових камер безпеки потребують принаймні одного кабелю або дроту для живлення; частина “бездротовий” часто відноситься саме до передачі даних. Однак деякі бездротові камери безпеки живляться від батареї, що робить їх справді бездротовими.

Структура систем відеоспостереження. У традиційній аналоговій системі відеоспостереження камери безпеки фіксують аналоговий відеосигнал і передають цей сигнал через коаксіальний кабель на цифровий відеореєстратор (DVR). Кожну камеру можна живити, підключивши джерело живлення безпосередньо до камери, або скориставшись сіамським кабелем RG59, який об’єднує відео та кабелі живлення. DVR перетворює аналоговий сигнал у цифровий, стискає його, а потім зберігає на жорсткому диску для подальшого пошуку. Інтелектуальні функції, вбудовані в DVR, забезпечують такі функції, як планування, виявлення руху та цифрове масштабування. Монітори для перегляду відео підключаються до DVR, або його можна налаштувати на публікацію у внутрішній мережі для перегляду на ПК. DVR також можна налаштувати на трансляцію через Інтернет і додати захист паролем та інші функції. При трансляції через Інтернет відео для всіх камер передається одним потоком (одна IP-адреса). Тому це дуже ефективно.

У світі IP кожна мережева камера фіксує аналогове зображення, але одразу перетворює його на цифрове всередині камери. Певна цифрова обробка може виконуватися прямо на камері, наприклад стиснення та виявлення руху. Потім цифровий відеопотік транслюється через локальну мережу (LAN) за допомогою кабелю Ethernet (CAT5 або CAT6). Структурна схема цифрової системи відеоспостереження зображена на рис. 1.

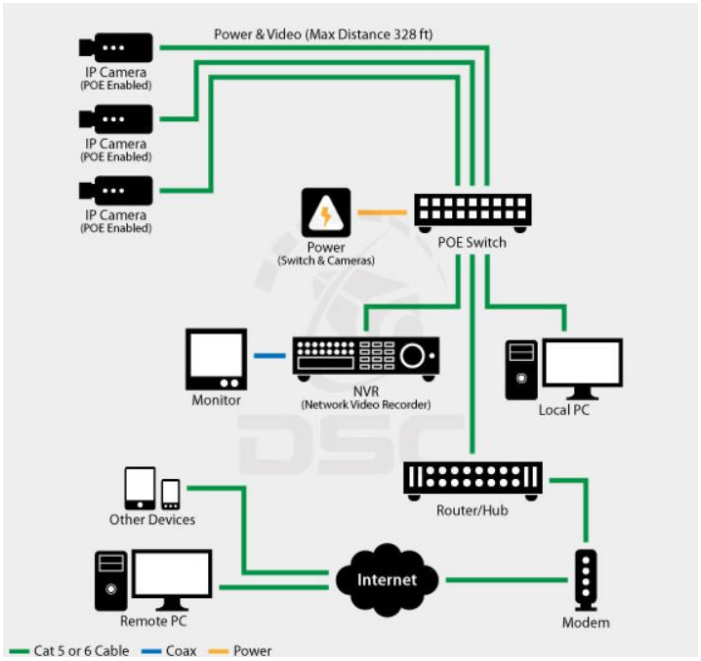


Рис.1. Структурна схема цифрової системи відеоспостереження

Живлення подається на камери через кабель Ethernet через адаптери Power-Over-Ethernet (POE), вбудовані в камери та на комутаторі (POE увімкнено). Кабель Ethernet для кожної камери під’єднується до комутатора, який підключається до мережевого концентратора. Як і для всіх мережевих пристроїв, для кожної мережевої камери потрібно виконати певні налаштування, щоб налаштувати її IP-адресу та інші ідентифікаційні атрибути. Класична побудова системи IP відеоспостереження. Цей варіант передбачає використання мережевого IP-відеореєстратора як сховища відеоінформації та керуючого пристрою, а також використання окремих джерел живлення для всіх відеокамер, які задіяні в системі. Розроблено та розглянуто класичну побудову системи IP-відеоспостереження.

Список використаних джерел

1. Бездротові камери спостереження: [Електронний ресурс]: – Режим доступу: https://nauka-online.com/wp-content/uploads/2018/07/Kucherenko.pdf

УДК 004.4

Баламут Д.О.
Вакалюк Т.А., зав.каф.інженерії програмного забезпечення, д-р пед. наук, проф.
Державний університет «Житомирська політехніка»

Інноваційні підходи до розробки та використання штучних нейронних мереж нового покоління

Штучні нейронні мережі нового покоління стали однією з найбільш перспективних галузей сучасної науки та технологій. Завдяки постійному прогресу в області обчислювальної техніки, а також зростанню обсягу доступних даних, штучні нейронні мережі стають все більш потужним інструментом для розв'язання різноманітних завдань у галузях, де вимагається інтелектуальна обробка інформації. Ця теза присвячена дослідженню та вдосконаленню штучних нейронних мереж нового покоління з метою вирішення складних завдань у галузі штучного інтелекту та машинного навчання. Зосереджуючись на останніх досягненнях у цій області, ми прагнемо розкрити потенціал нових алгоритмів, архітектур та підходів до навчання нейронних мереж, які дозволять покращити їх ефективність та швидкодіючість.

Технологія машинного навчання широко використовується в сучасному суспільстві, охоплюючи різні сфери від пошуку в Інтернеті та фільтрації контенту у соціальних мережах до надання рекомендацій на сайтах електронної комерції. Її вплив також відчутний в споживчих товарах, таких як камери та смартфони. Машинне навчання застосовується для розпізнавання об'єктів на зображеннях, перетворення мови в текст, аналізу новин, публікацій чи продуктів з урахуванням інтересів користувачів та надання відповідних результатів пошуку.

Глибоке навчання, в свою чергу, представляє собою галузь машинного навчання, яка базується на наборі алгоритмів, що намагаються моделювати високо абстрактні дані за допомогою кількох рівнів обробки з складними структурами або застосовують багато нелінійних перетворень. Глибоке навчання є частиною загального спектру методів машинного навчання, що базуються на представленні даних для навчання. [1]

Дані, такі як зображення, можуть бути представлені різними способами, такими як вектор значень інтенсивності для кожного пікселя або абстрактніше, у вигляді набору країв чи конкретних форм.Дослідження в цій галузі намагаються зробити кращі представлення та створити моделі для вивчення цих представлень із великомасштабних немаркованих даних. Деякі уявлення натхненні досягненнями в нейронауці та ґрунтуються на інтерпретаціях моделей обробки інформації та зв’язку в нервовій системі, таких як нейронне кодування, щоб спробувати визначити взаємозв’язки між різними стимулами та пов’язаними нейронними реакціями в мозку. [2]

Одним із аспектів глибокого навчання є заміщення ручно створених функцій алгоритмами, які ефективно працюють у неконтрольованому або напівконтрольованому навчанні, використовуючи ієрархічні функції.

Штучна нейронна мережа – це модель обробки інформації, яка моделює спосіб обробки інформації біологічними нейронними системами. Вона складається з великої кількості елементів, або нейронів, які взаємодіють між собою за допомогою зв'язків і працюють як єдина система для розв'язання конкретної задачі. Штучна нейронна мережа подібна до людського мозку, який здатний навчатися на основі досвіду, зберігати отримані знання і використовувати їх для прогнозування невідомих даних. Концепцію Штучна нейронна мережа вперше представили в 1943 році невролог Уоррен МакКалох та логік Уолтер Пітс. [2]

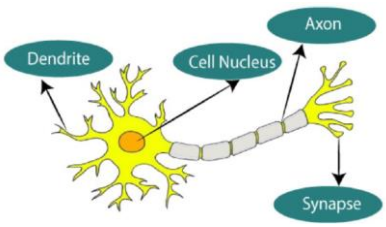


Рис. 1. Будова біологічного нейрона

Штучна нейронна мережа розглядається як система з трьох рівнів: вхідного, прихованого та вихідного, як показано на Рисунку 7. У цій структурі прихований рівень включає багато нейронів, які отримують вхідні дані від попередніх рівнів для подальшої обробки та трансформації цих даних перед передачею наступним рівням. Кількість прихованих рівнів може бути різною або вони можуть і взагалі відсутні. Кожен вузол у мережі називається нейроном, і кожен нейрон обробляє вхідні дані, повертаючи унікальний результат. Вихід одного нейрона може служити входом для інших нейронів, а ваги назначаються після визначення даних в вхідному рівні. Вагові коефіцієнти визначають важливість кожної змінної, де більші значення роблять більший внесок у результат. Усі вхідні дані помножуються на їх ваги, а потім сумуються.

Якщо отриманий вихід перевищує певний поріг, вузол активується, і дані передаються на наступний рівень мережі. Такий підхід дозволяє здійснювати передачу даних від одного вузла до іншого, де вихід одного вузла стає входом для наступного. Більшість штучних нейронних мереж працюють в режимі прямого проходження, тобто вони передають дані від входу до виходу лише по одному напрямку. Однак існують модифікації, які можуть використовувати зворотнє поширення, переносячи інформацію від виходу до входу, що поліпшує навчання мережі. [3]

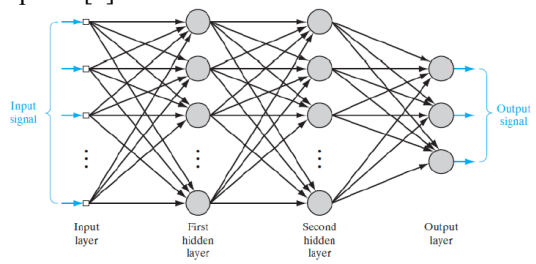


Рис. 2. Структура штучної нейронної мережі

Схоже до біологічних нейронів, кожен вхід штучного нейрона впливає на його вихід, і цей вплив регулюється ваговими коефіцієнтами, що присвоюються кожному входу (wi). Кожен коефіцієнт (wi) може мати позитивне або негативне значення, аналогічно до зв'язків у біологічних нейронах. Позитивне значення ваги (wi) відповідає збуджувальному впливу, тоді як негативне значення ваги відображає гальмівний вплив, подібно до біологічних нейронів. Тіло штучного нейрона відповідає за синтез вхідних сигналів для подальшої обробки та отримання вихідного результату. Цей процес обробки і обчислення буде розглянуто більш детально в наступному розділі. Вихід штучного нейрона подібний до аксона біологічного нейрона. Вихідний сигнал може бути розділений на кілька гілок у структурі подібній до дерева, щоб передати його як вхід для інших нейронів. [4]

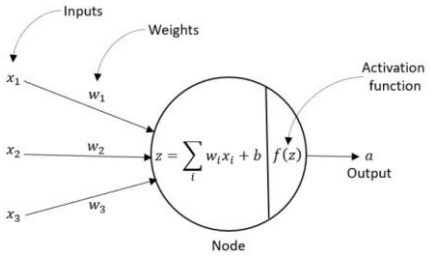


Рис. 3. Обробка нейрона в Штучна нейронна мережа

Алгоритм глибокого навчання функціонує за такою логікою: інформаційні потоки проходять через кілька рівнів аж до останнього рівня. Цей процес можна порівняти зі способом, яким людина вивчає та усвідомлює інформацію. На початкових етапах навчання фокус ставиться на конкретних поняттях, а з розвитком навичок глибокіші рівні використовують здобуті знання для більш глибокого дослідження та аналізу абстрактних концепцій. Процес створення представлення даних в глибокому навчанні включає в себе вилучення ознак. На початкових етапах система навчається розпізнавати більш конкретні аспекти інформації, а з часом, на більш глибоких рівнях, ці знання використовуються для аналізу та розуміння абстрактних понять. Ця складна архітектура глибокого навчання отримує потужність від глибоких нейронних мереж, які автоматично вилучають ознаки під час процесу навчання.

Розглянувши та дослідивши сучасні тенденції розвитку штучних нейронних мереж нового покоління, можна зробити важливі висновки. Штучні нейронні мережі стають все більш потужним та універсальним інструментом у галузі штучного інтелекту та машинного навчання. Їхні можливості розширюються завдяки постійному вдосконаленню алгоритмів, збільшенню обсягу доступних даних та розвитку обчислювальної техніки.

Список використаних джерел

1. Zhang, A, Lipton, Z, Li, M, Smola, A. 2021. "Exploring Deep Learning: A Comprehensive Guide." [Електронний ресурс] – Режим доступу до ресурсу: https://d2l.ai/d2l-en.pdf

2. Bhardwaj, N, Dixit, M. 2016. "An Overview of Facial Expression Detection: Techniques and Applications." International Journal of Signal Processing, Image Processing, and Pattern Recognition, Vol. 9, No. 6/2016, pp. 149 – 158.

3. Yang, G, Huang, T. 1992. "Detection of Human Faces in Challenging Backgrounds." Pattern Recognition, Vol. 27, No. 1/1994, pp. 53-63.

4. Yang, M, Kriegman, D, Ahuja, N. 2002. "Survey on Face Detection in Images." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1/2002.

УДК 004.738.5

Дзюба М. В., аспірант
Ільченко А.Л., аспірант
Чухов В.В., к.т.н., доц.
Державний університет «Житомирська політехніка»

Інтеграція даних гібридних систем охорони з IoT технологіями «розумного будинку»

Інтеграція гібридних систем охорони з технологіями «розумного будинку» на основі Інтернету речей (IoT) стає все більш актуальною, оскільки вона дозволяє створити більш комплексні та ефективні системи безпеки. Ця методика дає змогу об'єднати дані з різних джерел, таких як датчики руху, камери спостереження, сигналізація тривоги та інші IoT-пристрої, для забезпечення кращого захисту та контролю над житлом чи комерційною нерухомістю. Інтеграція гібридних систем охорони з IoT технологіями «розумного будинку» стає все більш актуальною, адже це дозволяє підвищити рівень безпеки та комфорту житла, оптимізувати використання ресурсів, зробити управління системою більш зручним та ефективним.

Існує декілька методів інтеграції даних гібридних систем охорони з IoT технологіями «розумного будинку»:

- використання API: багато систем охорони та IoT-пристроїв мають API (Application Programming Interface), які дозволяють розробникам програмного забезпечення інтегрувати їх з власними програмами або системами, що в свою чергу дозволяє IoT-обладнанню обмінюватися даними з іншими системами. Цей метод є гнучким та масштабованим, але він може потребувати знань програмування.
- використання протоколів шини: існують протоколи шини, такі як Modbus або BACnet, які дозволяють різним пристроям обмінюватися даними. Ці протоколи прості у використанні, але вони можуть бути менш гнучкими, ніж API;
- використання шлюзу: шлюз – це пристрій, який перетворює дані з одного формату в інший. Шлюзи можуть використовуватися для інтеграції систем охорони та IoT-пристроїв, які використовують різні протоколи;
- використання платформ IoT: багато виробників систем охорони та IoT-пристроїв пропонують платформи, які дозволяють об'єднувати дані з різних джерел та створювати правила автоматизації. Ці платформи зазвичай пропонують графічний інтерфейс користувача (GUI) для моніторингу та керування системою;
- використання хмарних сервісів: деякі системи охорони та IoT-пристроїв можуть підключатися до хмарних сервісів, які дозволяють їм обмінюватися даними з іншими системами. Цей метод простий у використанні, але він може потребувати фінансових затрат, оскільки необхідна підписка;
- використання MQTT: MQTT (Message Queuing Telemetry Transport) – це легкий протокол обміну повідомленнями, який часто використовується для IoT-пристроїв. Цей протокол може бути використаний для об'єднання даних з різних джерел та надсилання їх на центральний сервер.

Інтеграція гібридних систем охорони з IoT технологіями «розумного будинку» має багато переваг, зокрема:

- підвищена безпека: IoT-пристрої можуть використовуватися для виявлення та запобігання проникненню, а також для оповіщення про інші загрози, такі як пожежа або витік газу.
- підвищений комфорт: IoT-пристрої можуть використовуватися для автоматизації завдань, таких як регулювання температури, освітлення, музики тощо.
- оптимізація використання ресурсів: IoT-пристрої можуть використовуватися для збору даних про використання енергії та води з метою оптимізації використання ресурсів та економії коштів.
- зручне управління: системою охорони та IoT-пристроями можна управляти з одного центрального інтерфейсу, що робить управління більш зручним та ефективним.

При проєктуванні гібридної системи необхідно врахувати:

- безпека: інтеграція даних повинна бути безпечною, щоб запобігти несанкціонованому доступу до інформації;
- сумісність: всі системи та пристрої повинні бути сумісні один з одним;
- надійність: система повинна бути надійною та працювати без збоїв;
- масштабованість: система повинна мати можливість додавання нових IoT-пристроїв в подальшому

Інтеграція гібридних систем охорони з IoT технологіями «розумного будинку» стає все більш популярною, адже це дозволяє підвищити рівень безпеки, комфорту та ефективності житла. Вибір методу інтеграції та алгоритм впровадження залежить від конкретних потреб та вимог.

Список використаних джерел

1. Розумний будинок: [Електронний ресурс]: – Режим доступу: ttps://journals.uran.ua/atbp/article/download/32920/29533

УДК 004.4

Баламут Д.О.
Вакалюк Т.А., зав.каф.інженерії програмного забезпечення, д-р пед. наук, проф.
Державний університет «Житомирська політехніка»

Використання згорткових нейронних мереж для вдосконалення обробки зображень

У сучасному світі, зі сталим зростанням споживання енергії, виникає відчутна потреба в пошуку Сучасний світ машинного навчання та штучного інтелекту виявляє непередбачувану динаміку розвитку, визначеною переважно за рахунок новаційних методів та технологій. Одним із ключових напрямків у цій сфері є використання згорткових нейронних мереж (ЗНМ), які відіграють важливу роль у вдосконаленні обробки зображень та вирішенні завдань машинного бачення. ЗНМ здатні адаптуватися до складних структур даних та ефективно впоратися із завданнями аналізу великих обсягів інформації у реальному часі. У зв'язку з цим, виникає актуальна необхідність дослідження та вдосконалення ЗНМ для оптимізації їхнього функціонування та застосування в різноманітних сферах, що потребують аналізу зображень та взаємодії з навколишнім середовищем з використанням технологій машинного бачення.

Згорточна нейронна мережа (CNN) — це одна з передових моделей глибокого навчання, яка сьогодні допомагає дослідникам створювати інтелектуальні системи з високою точністю. Ця модель була розроблена та застосована до великих систем обробки зображень Facebook, Google або Amazon для різних цілей, таких як автоматичні алгоритми тегування, пошук зображень або рекомендації продуктів для споживачів. Згортка вперше була використана в цифровій обробці сигналів. Завдяки принципу перетворення інформації вчені застосували цю техніку для цифрової обробки зображень і відео. [1]

Це також означає, що між вхідним шаром і вузлом у наступному прихованому шарі потрібні 3072 ваги. Кількість ваг буде більше повторюватися, якщо кількість вузлів у прихованому шарі збільшується, а кількість прихованих шарів збільшується. Таким чином, маючи лише невелике зображення розміром 32x32, необхідна досить масивна модель нейронної мережі прямого зв’язку, що ускладнює роботу з більшими зображеннями. На основі цієї ідеї народилася згортка нейронної мережі зі структурою, відмінною від прямої нейронної мережі. Замість того, щоб усе зображення було безпосередньо підключено до вузла, лише локальна частина зображення підключена до вузла на наступному рівні. Вихідні дані зображення через шари моделі згорткової нейронної мережі вивчатимуть функції для виконання ефективної класифікації. [2]

Модель нейронної мережі народилася і широко застосовувалася для вирішення проблем розпізнавання. Однак для даних зображень нейронна мережа прямого зв’язку працює не дуже добре. Дані зображення досить великі, сіре зображення розміром 32x32 пікселя створить вектор ознак із 1024 вимірами, для кольорового зображення такого самого розміру буде 3072 виміри. (Симонян і Зіссерман 2014.)

В основному CNN включає такі рівні: згортковий рівень, рівень активації, рівень об’єднання (також відомий як рівень підвибірки) і повністю зв’язаний рівень. На рисунку 1 шари з’єднані між собою за допомогою механізму згортки. Наступний шар є результатом згортання попереднього шару, тому він здатний отримувати локальні з’єднання. Кожен нейрон наступного шару генерується з фільтрів, застосованих до локальної області зображення нейрона попереднього шару. Кожен такий шар піддається різним фільтрам, зазвичай від кількох сотень до кількох тисяч таких фільтрів. Інші класи, такі як pooling/subsam-pling, використовуються для отримання більш корисної інформації. Під час навчання згорточна нейронна мережа автоматично вивчає параметри для фільтрів. Як і в класифікації зображень, CNN намагатиметься знайти оптимальні параметри для відповідних фільтрів у порядку вихідний піксель > край > форма > обличчя > високий рівень ознак. Останній шар часто використовується для класифікації зображень. [2]

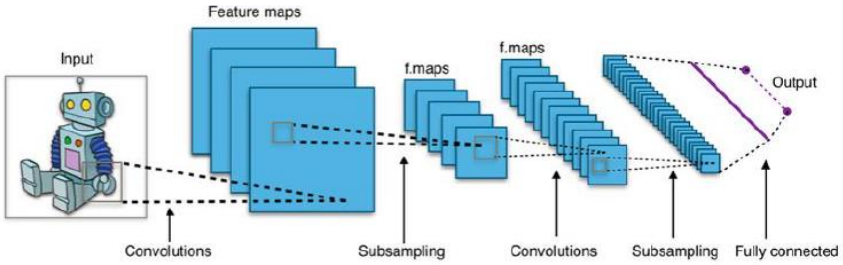


Рис. 1. Модель базових рівнів згорнутої мережі

На рисунку 2 видно, що використаний фільтр є матрицею розміром 3x3. Цей фільтр по черзі переміщається по кожній області зображення, доки не завершиться сканування всього зображення, створюючи нове зображення, розмір якого менше або дорівнює вхідному розміру. Таким чином, після надання вхідного зображення шару згортки, на виході буде серія зображень, що відповідають фільтрам,

використаним для виконання згортки. Ваги цих фільтрів уперше ініціалізуються випадковим чином і поступово покращуватимуться протягом процесу навчання. [3]

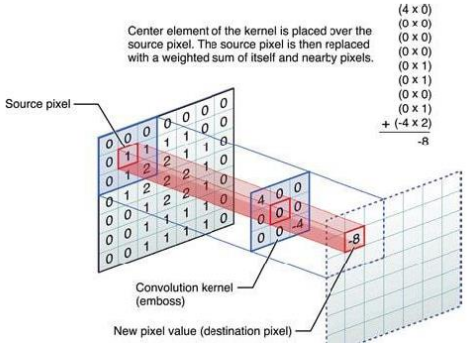


Рис. 2. Ілюстрація згортки на матриці зображення

Іншим основним обчислювальним компонентом у мережі CNN є об’єднання, яке зазвичай розміщується після шару згортки та шару ReLU, щоб зменшити розмір вихідного зображення, зберігаючи при цьому важливу інформацію вхідного зображення. Сьогодні існує два популярних методи вибірки: вибірка максимального значення пікселів (Max Pooling) і вибірка середнього значення пікселів у локальній області зображення. Таким чином, для кожного вхідного зображення, яке проходить вибірку, буде отримано відповідне вихідне зображення, розмір якого значно зменшується, але все ще зберігає необхідні характеристики для подальшого обчислення. [4]

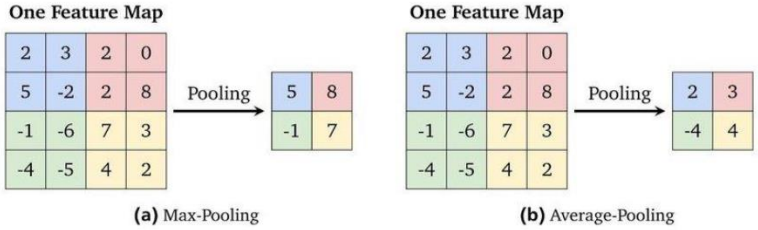


Рис. 3. Методи середнього об’єднання та максимального об’єднання

Цей повністю підключений рівень повністю схожий на традиційну нейронну мережу, тобто всі пікселі повністю підключені до вузла наступного рівня. Порівняно з традиційними нейронними мережами, вхідні зображення цього шару були значно зменшені в розмірі, але забезпечували важливу інформацію для ідентифікації. Таким чином, обчислення розпізнавання за допомогою моделі прямого зв’язку більше не є складним і трудомістким, як у традиційній нейронній мережі.

У результаті дослідження з використанням Python та бібліотек TensorFlow, Keras, OpenCV, NumPy та SciPy для аналізу та розпізнавання емоцій обличчя було виявлено високий потенціал цих інструментів у сферах штучного інтелекту та комп'ютерного зору. Застосування нейронних мереж та комп'ютерного зору дозволяє ефективно визначати емоційний стан особи на основі зображення обличчя. Це може мати важливе значення у різних галузях, таких як психологія, медицина, безпека та розваги. Розширення можливостей цих технологій може сприяти подальшому розвитку систем інтелектуального аналізу та взаємодії з людьми, забезпечуючи нові можливості для покращення якості життя та розвитку різних сфер діяльності.

Список використаних джерел

1. Zhang, A, Lipton, Z, Li, M, Smola, A. 2021. "Exploring Deep Learning: A Comprehensive Guide." [Електронний ресурс] – Режим доступу до ресурсу: https://d2l.ai/d2l-en.pdf

2. Bhardwaj, N, Dixit, M. 2016. "An Overview of Facial Expression Detection: Techniques and Applications." International Journal of Signal Processing, Image Processing, and Pattern Recognition, Vol. 9, No. 6/2016, pp. 149 – 158.

3. Yang, G, Huang, T. 1992. "Detection of Human Faces in Challenging Backgrounds." Pattern Recognition, Vol. 27, No. 1/1994, pp. 53-63.

4. Yang, M, Kriegman, D, Ahuja, N. 2002. "Survey on Face Detection in Images." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1/2002.5.

УДК 004

Вербовський О.Ю., студент, гр. ІПЗ-20-1, ФІКТ
Локтікова Т. М., ст. викл. кафедри ІПЗ
Лисогор Ю.І., ст. викл. кафедри ІПЗ
Державний університет «Житомирська політехніка»

Платформа для організації та виконання завдань з вбудованим чатом

На сьогоднішній день, в умовах війни, пост-пандемійного періоду та інших ситуацій, що впливають на рішення людей щодо зміни місця перебування на довгий термін, дуже складно здійснити навіть прості дії самотужки в місцях, в які немає можливості фізично потрапити, оскільки для цього потрібно витратити багато сил, енергії, коштів та інших ресурсів, щоби виконати завдання. Коли маєш близьких або друзів, які знаходяться на іншому кінці світу, привітати їх з особистою подією може стати справжнім викликом. Наприклад, привітати подругу з Днем народження в серпні букетом ромашок та смачним кавуном стає непростим завданням без допомоги місцевих друзів.

Даний проєкт має на меті полегшити та зробити більш доступними такі види комунікації та взаємодії в умовах глобальної нестабільності. Пропонована платформа дозволяє людям як віддалено обмінюватися реальними матеріальними подарунками, так і просто співати гарно пісні, розповідати виразно вірші, яких ще не бачив світ, або просто передавати пакет продуктів рідній бабусі, яку вже не бачив декілька років, не маючи можливості приїхати та допомогти.

Тож одним із найважливіших моментів програмного продукту є забезпечення можливості віртуальної присутності та активної участі у подіях, навіть якщо фізично люди знаходяться далеко один від одного. Така платформа сприятиме відновленню та зміцненню зв'язків між людьми в умовах сучасної нестабільності та важливості віддаленої спільності.

Для побудови застосунку обрано клієнт-серверну архітектуру, тому що вона має низку переваг, які відображають потреби платформи обміну послугами та завданнями між замовниками та виконавцями. По-перше, вона забезпечує ефективну взаємодію з сервером, передаючи або отримуючи дані лише за потребою, що дозволяє зменшити навантаження на мережу та підвищити ефективність обміну інформацією. По-друге, можливість асинхронної роботи дозволяє обробляти кілька запитів одночасно, покращуючи продуктивність та масштабованість системи. Відокремлення логіки серверу та інтерфейсу користувача спрощує розробку та супровід, що важливо для забезпечення гнучкості у процесі розробки та підтримки. Також архітектура сприяє гнучкості у розгортанні, дозволяючи легко створювати нові клієнтські застосунки, які задовольняють різні потреби користувачів.

У цьому проєкті для реалізації серверної частини була обрана платформа Node.js. Однією з ключових переваг Node.js є можливість виконувати код, написаний мовою JavaScript, що робить її кращим інструментом для створення масштабованих веб-застосунків. Крім того, використання асинхронності в Node.js дозволяє виконувати багато операцій одночасно, не очікуючи завершення кожної з них. Це особливо корисно для веб-серверів, які повинні ефективно обслуговувати багато запитів від користувачів одночасно. Такий підхід забезпечує швидший обмін даними між клієнтом і сервером, зменшуючи час очікування для користувача. Крім того, Node.js легко масштабується горизонтально, додаванням нових серверів для обробки збільшеного навантаження. Це дозволяє підтримувати високу доступність та швидкість системи при зростанні кількості користувачів, що надає можливість легко розширювати систему з ростом її популярності та користувацької бази.

Для розробки клієнтської частини проєкту була обрана бібліотека JavaScript – React. React є потужним інструментом, який дозволяє створювати високопродуктивні та ефективні односторінкові застосунки (Single Page Applications, SPA). React відзначається високою швидкодією, завдяки використанню віртуального DOM (Document Object Model) та ефективного алгоритму оновлення, що робить його особливо швидким у SPA. Крім того, бібліотека забезпечує зручну інтеграцію з серверною частиною, реалізованою на Node.js. Це забезпує гармонійну взаємодію між клієнтом та сервером на обох складових проєкту, сприяючи ефективній роботі та розвитку системи.

У даному проєкті для зберігання даних обрано MySQL. Обумовлено це тим, що MySQL є однією з найпоширених та добре вивчених систем керування реляційними базами даних. Її довгий період використання у великих проєктах та корпоративних системах свідчить про стабільність та надійність цієї системи. Крім цього, одним із найважливіших моментів є те, що MySQL є вільною та відкритою системою, доступною для використання без великих витрат. Це особливо важливо на початкових етапах розробки та для стартапів. Не менш важливою перевагою MySQL є інтеграція з серверними застосунками, написаними на Node.js, завдяки наявності бібліотек та драйверів для Node.js, а саме "mysql" або "sequelize". Ці інструменти дозволяють виконувати SQL-запити, отримувати та взаємодіяти з даними швидко, зручно і безпечно, що сприяє майбутньому оновленню та покращенню функціоналу платформи.

Для реалізації програмного продукту було спроєктовано базу даних.

ER-діаграма створеної бази даних представлена на рис. 1.

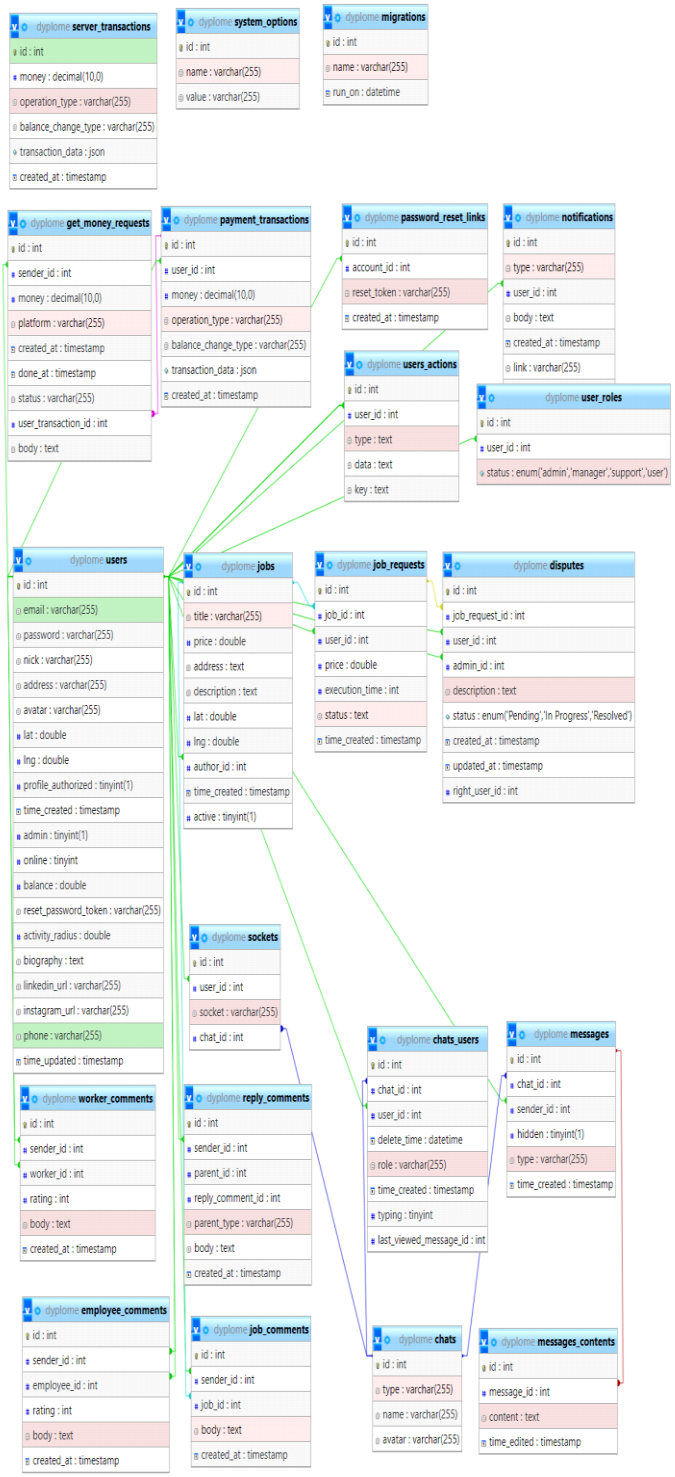


Рис.1. ER-діаграма створеної бази даних

Розроблений додаток проходить як функціональне, так і нефункціональне тестування. Це забезпечить відповідність програмного забезпечення всім вимогам та стандартам, що є ключовим аспектом у процесі розробки.

Список використаних джерел

1. Документація Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs/.
2.Документація React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn.
3. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/.

УДК 004

Костюченко А.В., студент, гр. ІПЗ-20-4
Кушнір Н.О., ст. викл. кафедри ІПЗ
Локтікова Т.М., ст. викл. кафедри ІПЗ
Державний університет «Житомирська політехніка»

Веб-сервіс для спільного перегляду фільмів та серіалів у режимі реального часу

У сучасному світі, із шаленим темпом розвитку технологій та загальнодоступною мережею Інтернет, відкриваються безліч нових можливостей у розробці різних веб-сервісів, одним із яких є сервіс спільного перегляду фільмів та серіалів. Цей феномен переходить на зовсім новий рівень завдяки високому розвитку веб-сервісів, які створюють враження спільного перебування у віртуальному кінозалі разом із друзями або навіть сторонніми користувачами з усього світу. Перегляд фільмів та серіалів перетворюється на справжню соціальну подію, коли глядачі можуть обмінюватися враженнями, коментарями та емоціями безпосередньо під час перегляду, ділитися своїми улюбленими моментами, висловлювати свої думки щодо сюжету та персонажів, обговорювати несподівані повороти сюжету та аналізувати різні аспекти фільмів.

Цілісний проєкт розпочинається з ідеї та концепції кінцевого продукту. Отже, спочатку визначаються мета і призначення системи, що розробляється. Веб-сервіс призначений для спільного перегляду фільмів та серіалів у реальному часі, надаючи можливість користувачам насолоджуватися вмістом одночасно та обмінюватися враженнями в онлайн-режимі. Кожен контент поділяється на категорії та жанри для зручності пошуку, а також можуть бути надані рекомендації на основі історії перегляду. Додатково, користувачі можуть створювати особисті кімнати перегляду та обговорювати враження в чаті під час перегляду.

Для побудови застосунку було обрано клієнт-серверну архітектуру із застосуванням веб-технологій, які не вимагають спеціалізованих знань або додаткових витрат на ліцензування, що спрощує розробку та впровадження проєкту.

Серверна частина розробленої системи базується на веб-сервері Node.js та базі даних MongoDB з використанням патернів MVC та REST API. Це спрощує організацію коду та забезпечує зручну взаємодію з клієнтською частиною. Дані зберігаються у вигляді документів, що забезпечує швидкий доступ до них.

Клієнтська частина розроблена як односторінковий веб-застосунок, який забезпечує можливість завантажувати лише ті елементи сторінки, які змінюються під час взаємодії користувача з ними. Це дозволяє зменшити час завантаження та підвищити швидкість роботи веб-застосунку.

Для створення сучасного і якісного веб-застосунку потрібно обрати оптимальний інструментарій для розробки. В ході проєктування бази даних було вирішено використовувати фреймворк Express.js, побудований на мові програмування JavaScript, який дозволяє забезпечити швидкий та ефективний обмін даними між клієнтом і сервером. Для забезпечення можливості взаємодії користувачів у реальному часі під час перегляду відеоконтенту, рішенням стало використання бібліотеки Socket.io, яка дозволяє передавати дані безпосередньо та миттєво між усіма під'єднаними користувачами. Для реалізації клієнтської частини було обрано відкриту бібліотеку JavaScript – React. React є потужним інструментом для створення односторінкових додатків (SPA) і має широку популярність у розробницькій спільноті.

Для реалізації програмного продукту було спроєктовано базу даних, яка містить 17 колекцій, головними з яких є:

1. Колекція Content – містить інформацію про відеоконтент (країни випуску, жанри, актори, режисери, списки, відгуки і т.д).
2. Колекція TypeContent – містить інформацію про тип відеоконтенту (назва типу, шлях і т.д).
3. Колекція Rooms – містить інформацію про кімнату (власник кімнати, обраний контент для перегляду, кількість під'єднаних користувачів, повідомлення і т.д).
4. Колекція Persons – містить інформацію про режисерів та акторів (вік, місце народження і т.д).
5. Колекція Seasons – являє собою колекцію сезонів відеоконтенту (назва сезону, епізоди сезону і т.д).
6. Колекція Episodes – містить інформацію про епізоди сезону відеоконтенту (назва епізоду, статус, дата випуску і т.д).

Список використаних джерел

1. Express Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/ HYPERLINK "https://expressjs.com/".
2. Socket.io Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://socket.io/docs/v4/.
3. React Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/

УДК 623.48

Мілевський О.В., аспірант
Манойлов В.П., к.т.н. проф.
Воротніков В.В., к.т.н. доц.
Державний університет "Житомирська політехніка"

Проблематика сучасних систем РНМ

Конкурентна боротьба останніх років десятиріччя призводить до мінімізації простоїв обладнання, підвищення швидкостей, зменшення обслуговуючого персоналу, ускладнення алгоритмів взаємодії всередині процесів. Сучасні вимоги до роботи обладнання, машин, систем, мереж, технологій, IoT та інших пристроїв є надвисокі. Тому в технічному обслуговуванні систем та обладнання використовуються нові методи та підходи, такі як метод прогнозованого стану (РНМ). Обладнання сигналізує про необхідність обслуговування лише у випадку потреби обслуговування; при цьому йде обслуговування лише того модуля або об'єктів, стан котрих почав погіршуватись й система контролю вбачає необхідність втручання. Метод РНМ виник в авіакосмічній галузі в 1990-х роках та поступово розповсюдився на найбільш відповідальні системи – системи гальм високошвидкісних потягів, управління гвинтокрилами, ударними винищувачами тощо. Наразі потенційні сфери застосування РНМ:

- промисловість: РНМ може використовуватися для прогнозування поломок обладнання на заводах, що допоможе запобігти простоям та зменшити витрати на ремонт.
- енергетика: РНМ може використовуватися для моніторингу стану електростанцій та ліній електропередач, що допоможе запобігти аваріям та покращити надійність електропостачання.
- авіація: РНМ може використовуватися для прогнозування поломок двигунів та інших компонентів літаків, що допоможе підвищити безпеку польотів.
- медицина: РНМ може використовуватися для моніторингу стану пацієнтів в лікарнях, що допоможе покращити діагностику та лікування.

Впроваджуючи складні системи, компанії визначають область застосування таких систем. Вартість та складність систем суттєво зменшують можливу зону застосувань. Все це справедливо і для методу прогнозованого стану (РНМ). Для прикладу, частота дискретизації для РНМ близько 100 кГц , кількість знятих даних лише за один оберт вентилятора або редуктора можуть досягати сотні тисяч. Зберігання цих даних потребує великих ресурсів пам'яті, а також швидкодіючих процесорів. Звісно ж, для сучасного настільного комп'ютера це не складна задача. Але мікроконтролери мають обмежену обчислювальну потужність і пам'ять. Використовування потужних ПЛК та написання програм РНМ для конкретної машини, потребує спеціальних знань від програміста. Це також суттєво гальмує розвиток РНМ. Тому для здешевлення прогнозованих систем, пішли шляхом розробки датчиків на основі MEMS, що включає мікроконтролер, де вся цифрова обробка сигналу виконувалася б датчиком.

Розробка датчиків на основі MEMS може стати одним із шляхів вирішення цих проблем. MEMS-датчики інтегрують мікроконтролер та цифрову обробку сигналу в один компактний корпус, що робить їх: дешевшими: виробництво MEMS-датчиків економніше, ніж окремих датчиків та процесорів; компактнішими: MEMS-датчики мають значно менші розміри, що робить їх зручнішими для розміщення на обладнанні; енергоефективнішими: MEMS-датчики споживають менше енергії, ніж традиційні датчики, що важливо для мобільних та автономних пристроїв; гнучкішими: MEMS-датчики можна програмувати та налаштовувати для виконання різних завдань, що розширює їхню сферу застосування.

Щоб виконувати обчислення датчиком, потрібно вдосконалювати також алгоритми. Найбільш розповсюджені алгоритми, котрі зазвичай використовуються в методах прогнозованого стану – швидке перетворення Фур'є (FFT), синхронне середнє значення в часі (TSA). Ці алгоритми успішно працюють в системах РНМ, але потребують потужних пристроїв для обчислень. Розробляючи датчики з вбудованими алгоритмами РНМ, намагаються покращити вищезгадані алгоритми (наприклад, використання алгоритма Кленшоу). Для прикладу, це дозволило реальний FFT пришвидшити в 14 разів відносно стандартного алгоритма FFT.

Таким чином, стало реальним виконувати обчислення алгоритму FFT датчиком на базі, наприклад, мікроконтролера AVR32UC30512C. Щодо алгоритму BEA, результати також були пришвидшені у 8 разів відносно стандартного, але цього не достатньо для використання відносно дешевих мікроконтролерів. Тому датчики вібродіагностики залишаються в дорогому сегменті, що затримує їх широке використання в РНМ. Розповсюдження систем раннього виявлення несправностей, до цього часу, не набуло широкомасштабного розгортання, оскільки потужностей дешевих лінійок мікропроцесорів недостатньо для обробки алгоритмів, а потужні процесори сильно здорожують систему. Тому потрібно вдосконалювати алгоритми, розробляти недорогі інтелектуальні датчики та системи оповіщення й зберігання даних.

УДК 621.317

Антонюк С.С., студент каф. КТуМтаТ гр.ТР-22-1
Корніюк А.В., ст. викладач каф. КТуМтаТ
Державний університет «Житомирська політехніка»

Система оповіщення про повітряну тривоги

З початку конфлікту на сході України у 2014 році, проблема забезпечення безпеки та ефективного оповіщення про небезпеку стала актуальною як ніколи. Як ми знаємо, 24 лютого 2022 року розпочалася агресія зі сторони росії. У зв'язку з цим, розвиток оповіщувальних систем став надзвичайно важливим завданням, особливо для місцевостей, де інфраструктура зазнала пошкоджень та прослуховування сигналу повітряної тривоги є неможливим.

На сьогоднішній день, одним з найефективніших рішень є використання мобільних застосунків та стаціонарних систем оповіщення. Що ж робити коли доступу до цих систем немає ? Наше рішення допомагає вирішити дану проблему. Всім нам відомі сайт alerts.in.ua та map.ukrainealarm.com. Їхні API дозволяюсь отримувати інформацію про тривоги та надсилати оповіщення користувачам через різні канали зв'язку, такі як SMS, мобільні додатки або голосові повідомлення.

Зупинимось детальніше на alerts.in.ua. Отримати доступ до API сайту alerts.in.ua можна подавши заявку на їхньому веб-сайті та отримавши ключ доступу. Після цього можна розробити програмне забезпечення, яке буде використовувати цей ключ для взаємодії з API та розсилання оповіщень користувачам. Сам сайт також містить документацію для використання їхнього API, включаючи приклади коду на мові програмування Python та Ruby, який можна використовувати для визначення тієї чи іншої інформації як: список активних тривог, статус повітряних тривог в областях, визначення статусу тривоги у певній області, визначення історії тривог за певний період. Ця документація також містить унікальні ідентифікатори для кожної області та міста зі спеціальним статусом. Крім того, можна використовувати посилання URL для доступу до статусу тривоги в конкретних областях.

Як приклад розглянемо використання url адреси:

https://api.alerts.in.ua/v1/iot/active_air_raid_alerts/10.json?token=(ваш токен) у даному випадку ця адреса буде відображати стан тривоги в області з ID номер 10, в документації це Житомирська область. При заході на сайт, ми будемо отримувати три коди значення: A – повітряна тривога активна в області, P – часткова тривога в районах чи громадах, N – немає інформації про повітряну тривогу.

Для створення системи сповіщення потрібно врахувати кілька ключових компонентів. Перш за все, потрібно вибрати мікроконтролер, який буде керувати системою. Можна використати STM32, Arduino або ESP32(якщо є доступ до Wi - Fi). Далі потрібен доступ до інтернету, з цим нам може допомогти модуль SIM800, якщо немає доступу до Wi – Fi. Також для роботи пристрою необхідне певне живлення, потрібно щоб його вистачило на 24 години. Тобто у нашого мікроконтролера повинна бути містка батарея для живлення, а також зарядний пристрій. З базовою конфігурацією визначилися, але ще нам необхідний гучномовець, який буде генерувати певний звук який буде підсилений підсилювачем звуку.

Після вибору мікроконтролера, потрібно розробити програмне забезпечення, яке буде отримувати дані з API сайту alerts.in.ua та аналізувати їх. У випадку отримання сигналу про небезпеку, програма може активувати генератор звукових сигналів для оповіщення користувачів. У нас є два варіанти, якщо ми будемо використовувати ESP32 ми повинні підключити його до Wi – Fi мережі. Далі ми повинні написати код, який буде виконувати HTTP запити до API сайту alerts.in.ua, отримуючи дані про повітряну тривогу. Потім ці запити повинні будуть оброблюватися і генеруватимуть певний звуковий сигнал, який ми будемо підсилювати і він звучатиме на рупорний гучномовець.

Другий варіант, використання модуля GSM SIM800 та мікроконтролера STM32 за допомогою UART портів. Цей варіант трохи складніший, адже необхідно мати добре підключення до мережі. Спочатку ми встановлюємо сім картку в модуль зв’язку певного оператору і налаштовуємо її. Далі, так як ми використовуємо STM32 код потрібно писати на мові програмування C/C++. Потім пишемо програмне забезпечення, яке буде мати доступ до API сайту та надсилатиме запити на перевірку стану повітряної тривоги в тій чи іншій області. Тут також можна використовувати вбудовану бібліотеку для роботи з HTTP запитами. Після того як програма спрацьовує вона повинна перероблювати цей сигнал в звуковий, після цього в роботу вступає підсилювач звуку завдяки якому ми будемо чути повітряну тривогу на рупорному гучномовці з максимальною гучністю.

Саму структурну схему даного пристрою можна глянути на рис. 1 нижче.

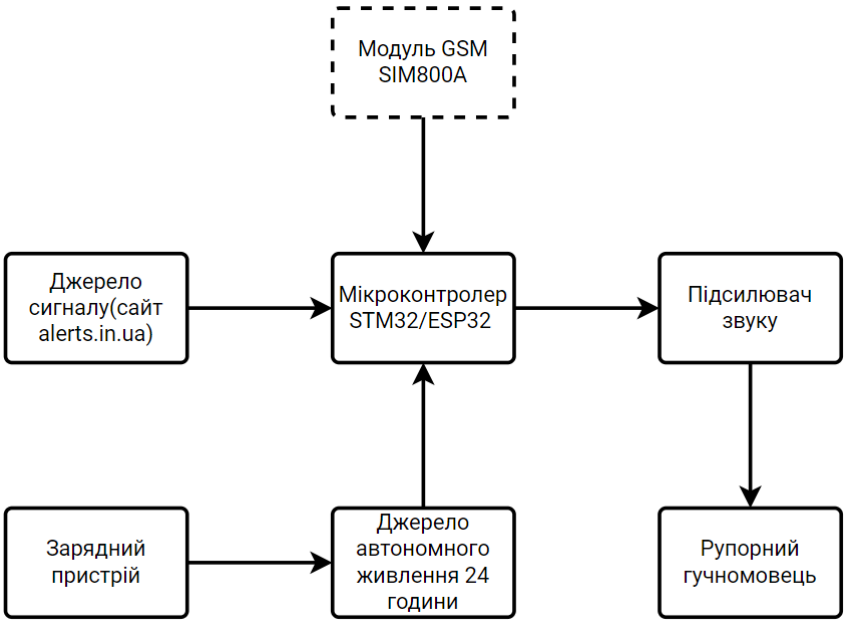


Рис. 1. Структурна схема пристрою сповіщення

Створення оповіщувальної системи на основі API сайту alerts.in.ua представляє собою значущий крок в напрямку забезпечення безпеки та захисту населення. Ця інноваційна ініціатива має великий потенціал для оптимізації процесів надання інформації про потенційні небезпеки та надзвичайні події, що відбуваються.

Основними перевагами такої системи є можливість своєчасного оповіщення громадян про небезпеку, що дозволяє ефективно взаємодіяти з різними видами ризиків та подій. Крім того, дана система може бути адаптована для використання у звичайних умовах для оповіщення мешканців сіл та міст про рутинні події або важливі оголошення. Додатково, розробка системи оповіщення про повітряну тривогу для віддалених або недоступних територій є критично важливою у контексті забезпечення безпеки населення та захисту довкілля в умовах можливого забруднення або інших атмосферних небезпек.

Науковий аналіз впровадження такої системи показує, що вона може значно підвищити рівень захисту населення, знизити потенційні ризики для здоров'я та майна громадян та сприяти збільшенню свідомості про потенційні небезпеки. Крім того, ця ініціатива може сприяти підвищенню готовності населення до реагування на негативні ситуації, забезпечуючи ефективну взаємодію між владою та громадськістю в умовах кризових ситуацій. Успішне впровадження та функціонування такої системи може мати значущий вплив на загальний рівень безпеки в суспільстві та сприяти запобіганню потенційних негативних наслідків для здоров'я населення та навколишнього середовища.

Отже, розробка та вдосконалення таких систем на базі передових технологій представляє собою перспективний напрямок в галузі забезпечення безпеки населення в умовах, коли традиційні методи сповіщення можуть бути недостатні або неефективні.

Список використаних джерел

1. Мапа тривог України. Карта повітряних тривог України. URL: http://www.alerts.in.ua.
2. STM32 32-bit Arm Cortex MCUs. STMicroelectronics. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.
3. ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems. Wireless SoCs, Software, Cloud and AIoT Solutions | Espressif Systems. URL: http://www.espressif.com/en/products/socs/esp32
4. Norris D. Programming with STM32: Getting Started with the Nucleo Board and C/C++. McGraw-Hill Education TAB, 2018. 304 p.
5. Cameron N. Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and Wi-Fi Enabled Devices. Apress L. P., 2020.

УДК 623.48

Сімчук А.Р., аспірант
Мацієвський В.А., аспірант
Нікітчук Т.М., к.т.н., доц.
Державний університет «Житомирська політехніка»

Засоби передачі медичної інформації у інформаційно-телекомунікаційних системах

Телекомунікаційні системи являють собою комплекс програмного та апаратного обладнання, яке з'єднане один з одним в один ланцюг, що здійснює передачу даних з однієї точки в іншу. Така передача даних можлива завдяки чіткій структуризації телекомунікаційної мережі. Текстова, голосова, графічна, відео- та аудіо інформація – далеко не повний перелік можливостей будь-якої телекомунікаційної системи.

Інформаційно-телекомунікаційні системи в даний час отримали широку популярність – мережі стали більш потужними, комутація пакетів даних підвищилася, з'явилися оптичні телекомунікаційні системи тощо.

У телекомунікаційних системах для передачі медичної інформації використовується широкий спектр технологій. Деякі з найпоширеніших включають:

- **Проводовий зв’язок.** Це традиційний метод передачі даних, який використовує телефонні лінії або виділені лінії для з'єднання двох точок. Проводовий зв'язок надійний та безпечний, але він може бути дорогим та складним у налаштуванні.
- **Бездротові зв'язок.** Для передачі інформації може використовуватися інфрачервоне випромінювання, радіохвилі, оптичне або лазерне випромінювання. На сьогодні існує безліч бездротових технологій, відомих користувачам по їхніх маркетингових назвах, таким як Wi-Fi, WiMAX, Bluetooth та інші. Кожна технологія має певні характеристики, які визначають її область застосування. Бездротовий зв'язок зручний та портативний, але менш надійний, ніж провідний.
- **Мережі мобільного зв'язку.** Мобільні мережі можна використовувати для передачі медичної інформації з мобільних пристроїв, таких як смартфони та планшети. Мобільні мережі зручні та доступні, але вони можуть бути менш надійними, ніж інші типи мереж, та мають обмежену пропускну здатність.
- **Мережі на основі IP:** Мережі на основі IP, такі як Інтернет, також можна використовувати для передачі медичної інформації. Мережі на основі IP зручні та масштабовані, але вони можуть бути менш надійними, ніж інші типи мереж, – вразливі до кібератак.

Вибір методу передачі даних для медичної інформації залежить від низки факторів, включаючи:

- **Тип медичної інформації:** деякі типи медичної інформації, такі як рентгенівські знімки, потребують великої пропускної здатності, тоді як інші, такі як електронні медичні записи, можуть бути передані з меншою пропускною здатністю.
- **Рівень безпеки:** медична інформація є конфіденційною, тому важливо використовувати безпечний метод передачі даних.
- **Вартість:** вартість методу передачі даних є важливим фактором, особливо для великих організацій.
- **Доступність:** метод передачі даних повинен бути доступний користувачам, яким він потрібен.

Важливо зазначити, що жоден метод передачі даних не є досконалим. Важливо використовувати комбінацію методів для забезпечення надійної, безпечної та доступної передачі медичної інформації.

Окрім вищезазначених методів, для передачі медичної інформації у телекомунікаційних системах також використовуються інші технології, такі як:

- **Супутниковий зв'язок:** супутниковий зв'язок можна використовувати для передачі медичної інформації в райони, де немає доступу до інших типів мереж.
- **Телемедицина:** телемедицина – використання телекомунікаційних технологій для надання медичних послуг віддаленим пацієнтам.
- **Здоров'я mHealth:** mHealth – це використання мобільних пристроїв для надання та отримання медичних послуг.

Нові технології зв’язку нашого століття – наскрізна передача від датчика до хмари. Якщо раніше через Ethernet об’єднувалися лише окремі пристрої, то зараз «виростають» цілі мережі. У процесі оцифрування та впровадження Industrie 4.0 та IoT в майбутньому буде підключено ще більше крайових пристроїв. Крім даних для керування, вони також передаватимуть інформацію за потребою до різних адресатів у мережі чи в хмарі. Для керування такими обсягами даних і забезпечення можливості використання критичних застосунків у режимі реального часу потрібні нові технології, такі як TSN, SPE, APL і 5G.

УДК 004.4

Баламут Д.О.
Вакалюк Т.А., зав.каф.інженерії програмного забезпечення, д-р пед. наук, проф
Державний університет «Житомирська політехніка»

Python та його бібліотеки: аналіз за виразами обличчя

Визначення осіб є завданням, що включає синтез різних компонентів. Для цього необхідні ключові модулі, такі як визначення положення обличчя, виділення характерних рис та класифікація. Ці етапи дозволяють встановити емоційний стан людини на зображенні. В галузі обробки зображень відбувається активне дослідження та розвиток відомих наукових центрів, університетів і академій. Одним із напрямків, який наростає в популярності, є розпізнавання та класифікація зображень. Основна концепція полягає в аналізі зображень на основі інформації, отриманої від датчиків, таких як камери і веб-камери. Застосування технологій класифікації в даний момент активно розвивається в різних сферах, таких як наука, бізнес, безпека, охорона здоров'я, а також у соціальних дослідженнях, політиці, уряді та інших галузях. Організації, які мають великі обсяги неструктурованих даних, знаходять полегшення в обробці, якщо ці дані нормалізуються за темами або мітками. Технічною основою для розв'язання завдань класифікації служать штучний інтелект (AI) і глибоке навчання (DL).

Поряд з іншими формами розпізнавання, такими як розпізнавання мови, почерку, відбитків пальців, сітківки ока, проблема розпізнавання емоцій на людських обличчях привертає увагу найбільш допитливої групи вчених. Завдяки кращим знанням про емоції на обличчі люди можуть розрізняти негативний і позитивний зворотний зв’язок, наприклад, в академічному середовищі, де вчителі/професори можуть визначити, чи їхні учасники розуміють поточну тему, а також у сфері охорони здоров’я, де психологи можуть швидко надати терапію. сеанс для пацієнтів із сильною тривогою, які не люблять прямого зорового контакту. Емоційний аналіз оптимально використовується в онлайн-додатках з точки зору клієнтів як спосіб зібрати думки з різних досліджень [1].

З розповсюдженням камер спостереження в аеропортах, на робочих місцях, у навчальних закладах, банкоматах та банках, розпізнавання виразу обличчя (FER) стає все більш важливою частиною взаємодії людини з машиною. FER також може використовуватися в дослідженнях поведінкової психології, обслуговуванні клієнтів та системах рекомендацій на основі зображень. Вираз обличчя відображає настрій або емоційний стан людини в певний момент часу, наприклад, смуток, радість або гнів. Пол Екман виділив шість основних емоцій: смуток, радість, гнів, страх, огиду та здивування. Розпізнавання обличчя є першим кроком до розпізнавання емоцій. На цьому етапі обличчя ідентифікується на зображенні, а інші об'єкти (якщо такі є) видаляються. Після ідентифікації людського обличчя виділяються його характерні риси та формується їх представлення.

На основі отриманих ознак емоції класифікуються за однією з шести основних категорій, як показано на рис.1. Існує багато досліджень, спрямованих на підвищення точності розпізнавання емоцій.

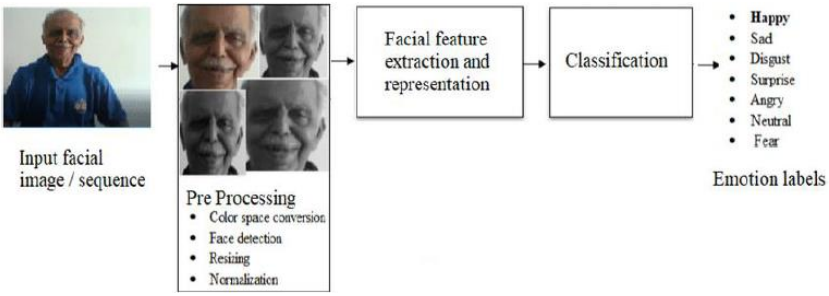


Рис.1. Основні етапи автоматизованого FER

Розпізнавання виразів обличчя (FER) використовує два основних підходи: аналіз зовнішності та геометричний аналіз. Аналіз зовнішності фокусується на інформації, отриманій з інтенсивності пікселів або цілого зображення. Для цього використовуються трансформації, фільтри, машинне навчання, статистичні дані та інші методи. Геометричний аналіз досліджує форму, відстань та розташування компонентів обличчя, таких як лицьові м'язи, очі, рот, лоб тощо. Психолог Пол Екман зробив значний внесок у цю сферу, розробивши в 1978 році Систему кодування дій на обличчі (FACS). FACS використовується для моніторингу та стандартизації емоцій, що виражаються на обличчі. Система кодує вирази обличчя за допомогою описів рухів м'язів (AU), які можна комбінувати для опису складніших емоцій.

Розпізнавання виразів обличчя (FER) використовує два основних підходи: аналіз зовнішності та геометричний аналіз. Аналіз зовнішності фокусується на інформації, отриманій з інтенсивності пікселів

або цілого зображення. Для цього використовуються трансформації, фільтри, машинне навчання, статистичні дані та інші методи. Геометричний аналіз досліджує форму, відстань та розташування компонентів обличчя, таких як лицьові м'язи, очі, рот, лоб тощо. Психолог Пол Екман зробив значний внесок у цю сферу, розробивши в 1978 році Систему кодування дій на обличчі (FACS). FACS використовується для моніторингу та стандартизації емоцій, що виражаються на обличчі. Система кодує вирази обличчя за допомогою описів рухів м'язів (AU), які можна комбінувати для опису складніших емоцій.

Шінха використовує невеликий набір інваріантів зображення в області зображення для опису простору зразків. Основна концепція ґрунтується на варіаціях рівнів яскравості різних ділянок обличчя, таких як очі, щоки та лоб, при цьому співвідношення рівнів яскравості інших областей залишається практично сталим [5]. Визначення пар співвідношення рівнів яскравості для конкретної області (де частина темніша або світліша) забезпечує ефективну інваріантність. Ділянки з рівномірною яскравістю розглядаються як пропорційний відбір, що представляє грубий вибір в просторі зображення обличчя, з основними рисами, такими як очі, щоки та лоб. Система зберігає зміни яскравості областей обличчя в відповідному наборі, утворюючи пари світліше-темніше між субобластями. Визначення обличчя відбувається тоді, коли зображення збігається з усіма парами світліше-темніше. [4]

У методі "Відповідність шаблону" стандартні образи облич, зазвичай, представляють собою прямокутні обличчя, заздалегідь визначені або параметризовані через функції. Система, базуючись на вхідному зображенні, обчислює значення кореляції зі стандартними зразками для контурів обличчя, очей, носа та рота. Автори вирішують наявність обличчя на зображенні чи відсутність, використовуючи ці значення кореляції. Цей метод має певні переваги, такі як простота налаштування, але його ефективність обмежена в разі змін у пропорціях, позах та формах [4].

Ця концепція виникла з різниці в інтенсивності між локальними сусідствами, яка потім була розширена за допомогою Wavelet Transform для розпізнавання пішоходів, ідентифікації автомобілів та виявлення облич. [6] На Рисунку 4 виокремлено видимий закономірність у 23 визначених зв'язках. Застосовуючи ці відношення для класифікації, виділяється 11 значущих (чорні стрілки) та 12 правильних (сірі стрілки) зв'язків. Кожна стрілка представляє відношення. Зв'язок, який відповідає шаблону обличчя, визначається, коли відношення між двома областями перевищує певне порогове значення. Ці зв'язки вважаються ідентифікацією обличчя, якщо вони перевищують порогове значення. Метод порівняння порядку шаблону для розпізнавання людських обличчів був представлений Мяо. На першому етапі зображення обертається від -200 до 20 з кроком 50 в порядку. [7] Створюються зображення з різною роздільною здатністю, а потім використовуються математика Лапласа для визначення контурів. Зображення обличчя з контуром відображає шість компонентів: дві брови, два очі, ніс і рот.

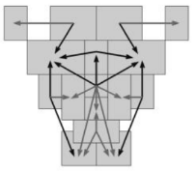


Рис. 2. Шаблон обличчя з 16 регіонами та 23 зв’язками (стрілки), заснований на методі Синха

Це підходить зверху вниз, де визначаються основні правила для опису характеристик обличчя та їх взаємозв'язків. Наприклад, обличчя зазвичай має два очі, які симетричні відносно вертикальної осі посередині обличчя, і присутні ніс і рот. Зв'язки між цими характеристиками можна виразити через відстані та положення. Зазвичай автори визначають риси обличчя, щоб отримати кандидатів, а потім застосовують правила для ідентифікації того, які кандидати є обличчями, а які - ні.

Проте, використання цього підходу ускладнюється ефективним перекладом людських знань у формалізовані правила. Якщо правила занадто деталізовані, можна пропустити визначення облич на зображенні, оскільки не всі обличчя можуть відповідати усім визначеним правилам. З іншого боку, занадто загальні правила можуть призвести до помилкової ідентифікації областей, які не є обличчями, або, навпаки, можуть пропустити деякі обличчя. Також важко розширити вимоги задачі для визначення обличчя в різних позах.

Рис. 3. Вихідне зображення (a) n = 1 і відповідні зображення з низькою роздільною здатністю (b) n = 4, (c) n = 8, (d) n = 16

Ян і Хуан використовують підхід, що слідує описаному вище для ідентифікації обличчя [3]. Їх система складається з трьох рівнів правил. На найвищому рівні вони використовують вікно сканування на зображенні та передають набір правил для пошуку можливих кандидатів на обличчя. На наступному рівні вони використовують правила для опису загальної форми обличчя. На останньому рівні використовують інший набір правил для детального аналізу рис обличчя.

Використовується впорядкована система з різною роздільною здатністю, як показано на Рисунку 4. Правила верхнього рівня для пошуку кандидатів включають "центр обличчя (темніша частина на Рисунку 3) має чотири частини з однією основною рівністю", "верхня периферія обличчя (світліша частина на Рисунку 3) має певну основну рівність" і "ступінь різниці між середніми значеннями сірого медіани в центрі та верхньої огинавної частини є значущою". На другому рівні система перевіряє гістограми кандидатів для виключення тих, які не є обличчями, і визначення меж навколо кандидата. На останньому рівні решта кандидатів перевіряють риси обличчя з точки зору очей і рота.

Вони застосовують стратегію "від грубого до тонкого" або "поступове прояснення", щоб оптимізувати обчислення під час обробки. Незважаючи на невисокий рівень точності, це є передумовою для подальших досліджень [4].

Рис. 4. Типовий метод дослідження обличчя, заснований на знаннях

У цьому методі шаблони вивчаються на основі прикладів фотографій, відмінності від підходів, які порівнюють шаблони з передвизначеними експертами шаблонами. Ці методи часто використовують статистичні, імовірнісні та методи машинного навчання для ідентифікації відповідних аспектів як на обличчях, так і в них. Отримані ознаки можна використовувати для розпізнавання людських облич за допомогою вивчених функцій, таких як дискримінантні функції або моделі розподілу. Проблема зменшення кількості вимірів часто обумовлена необхідністю підвищення ефективності обчислень та ефективності прийняття рішень. [4] Зображення або вектор ознак, отриманий із зображення, розглядається як випадкова величина x. Випадкова змінна x характеризується як обличчя чи не обличчя згідно з формулою, обчисленою на основі умовних функцій щільності класифікації: p(x | обличчя) і p(x | не обличчя). Для класифікації кандидата як обличчя чи не обличчя можуть бути використані Байєсівські класифікатори або класифікатори максимальної ймовірності.

Пряме використання класифікатора Байєса ускладнене через високу розмірність x. Функції p(x | обличчя) і p(x | не обличчя) представлені поліномами, і їх складно розуміти, побудовуючи форми параметрів або природного оцифрування. Багато досліджень в цьому напрямку зосереджені на пошуку параметричних або непараметричних наближень для p(x | обличчя) і p(x | не обличчя). У рамках методу на основі обличчя включає пошук дискримінантної функції, такої як площина рішення чи гіперплощина для розділення даних. Використовується порогова функція для розрізнення двох класів даних: обличчя та шаблони без обличчя. Зразки зображень проектуються в простір з меншою кількістю вимірів, а для класифікації використовується дискримінантна функція, ґрунтована на вимірюванні відстані. Також може бути побудована нелінійна поверхня прийняття рішень, використовуючи нейронну мережу з багатьма шарами.

Python та його бібліотеки виявляються потужними інструментами для аналізу за виразами обличчя. З використанням спеціалізованих бібліотек, таких як OpenCV, Dlib та TensorFlow, можна ефективно визначати різні емоції на обличчі, виконувати відстеження рухів та реалізовувати інші завдання з обробки зображень. Це особливо важливо у сферах, де важливо враховувати емоційний стан користувачів, таких як реклама, медицина або психологія. Такий аналіз дозволяє отримувати цінні дані для подальшого аналізу та прийняття рішень, що робить Python і його бібліотеки незамінними інструментами у розробці програм та дослідженнях з цієї області.

Список використаних джерел

1. Zhang, A, Lipton, Z, Li, M, Smola, A. 2021. "Exploring Deep Learning: A Comprehensive Guide." [Електронний ресурс] – Режим доступу до ресурсу: https://d2l.ai/d2l-en.pdf

2. Bhardwaj, N, Dixit, M. 2016. "An Overview of Facial Expression Detection: Techniques and Applications." International Journal of Signal Processing, Image Processing, and Pattern Recognition, Vol. 9, No. 6/2016, pp. 149 – 158.

3. Yang, G, Huang, T. 1992. "Detection of Human Faces in Challenging Backgrounds." Pattern Recognition, Vol. 27, No. 1/1994, pp. 53-63.

4. Yang, M, Kriegman, D, Ahuja, N. 2002. "Survey on Face Detection in Images." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1/2002.

УДК 004.4

Мельниченко Д.В.
Вакалюк Т.А., зав.каф.інженерії програмного забезпечення, д-р пед. наук, проф.
Державний університет «Житомирська політехніка»

Веб застосунок для керування робочим часом: огляд аналогів

В сучасному світі наявність програмних продуктів у будь-яких сферах діяльності значно покращує умови роботи та ефективність процесів. Не виключенням для цього твердження є інструменти, які допомагають компаніям автоматизувати велику кількість мануальної роботи. Однією із ключових частин планування програмного продукту є аналіз аналогічних розробок. Ця робота допоможе визначити певні особливості, функціонал та вимоги до продукту. Крім того, це також і пошук ідей, які можна втілити у власному проекті. Також можна виділити недоліки в аналогах, які було б добре виправити у власній розробці.

Після проведення пошуку аналогічних розробок, було знайдено велику кількість програмних продуктів для порівняння. Нажаль, більшість з них потребують замовлення демо програми від лиця компанії, тому неможливо запустити ці сервісу власноруч. Отже, будемо робити аналіз покладаючись на опис самих розробок та знайденої про них інформації в мережі.

Аналогічні системи мають дуже широкий список функціоналу, який розроблюється великими командами протягом багатьох років. Вони надають змогу клієнтам автоматизувати майже усі процеси управління персоналом. Тому основну увагу будемо звертати на окремі частини аналогічних розробок, також їх загальний стан.

Для аналізу аналогічних розробок було виділено три сервіси HiBob, PeopleForce та Hurma.

Платформа Hibob, орієнтована на дизайн [1] — це рішення для управління людськими ресурсами (HRM), створене для покращення досвіду роботи співробітників із зосередженням на малому та середньому бізнесі (рис. 1). За словами розробника, bob поєднує в собі можливості управління людським капіталом із адаптивною функціональністю інтерфейсу користувача. Платформа спрощує адаптацію, відвідуваність і перевірку ефективності для фахівців з управління персоналом і співробітників як користувачів за допомогою настроюваних інформаційних панелей і звітів, які спрощують документообіг адміністратора.

До переваг віднесемо:

- Дуже простий, стильний і зручний дизайн за яким ховається багато зручних функцій.
- Величезна кількість функціоналу що підтримується системою.
- Гарно пропрацьовані введення та виведення працівника в компанію.
- Ефективне відстеження даних про співробітників, їх аналітика та представлення.
- Чудова система управління відсутністю працівника.
- Висока адаптивність під потреби різних компаній.
- Можливість збирати й оцінювати комплексний відгук, покращуючи процеси управління продуктивністю.
- Користувачі можуть налаштувати робочі процеси в системі, які дозволяють адаптувати її до потреб організації.
- Вбудована система управління документами.

До недоліків слід віднести:

- Деякі користувачі мають проблеми із інтеграцією з іншими системами.
- Обмеження в управлінні завданнями.
- Обмежені статусом життєвого циклу робітників запобігає великій адаптивності до різної політики та робочих процесів компаній.
- Велика кількість недопрацювань під час роботи з електронним підписом.

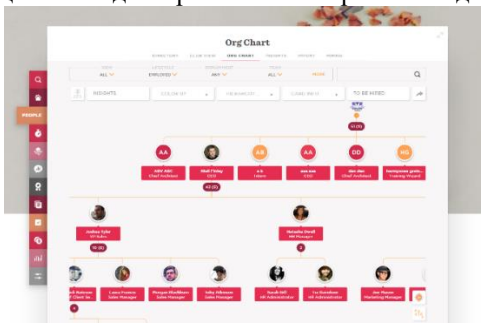


Рис. 1. Інтерфейс системи HiBob

PeopleForce [2] - це хмарна система управління людськими ресурсами (HRMS), призначена для допомоги підприємствам оптимізувати процеси управління людськими ресурсами на кожному етапі життєвого циклу співробітника: від найму і онбордингу до залучення, відстеження робочого часу та оцінки результативності через єдину платформу (рис. 2). Це дозволяє командам у сфері управління людськими ресурсами зберігати та відстежувати дані про співробітників в централізованому репозиторії, вести облік заявок від кандидатів та проводити опитування для аналізу задоволеності співробітників.

PeopleForce надає API, що дозволяє підприємствам інтегрувати систему з різними платформами сторонніх розробників, такими як Telegram, Slack, LinkedIn, Jooble, Work.ua, G Suite та інші. Інші функції включають управління користувачами, експорт даних, аналітику, управління завданнями, відстеження відсутності, контроль доступу та автоматизовані нагадування.

До переваг відносимо:

- Зручний для користувача інтерфейс.
- Велика кількість локалізації системи.
- Гарний баланс між ціною та функціоналом.
- Відносно легке налаштування відповідно до політики компанії.
- Автоматичне завантаження резюме робітника через підключення до багатьох різних сайтів, де робітник може бути зареєстрованим.
- Багато зручних функцій збирання, аналізу та формування звітності.
- Система прогнозування робочого навантаження, вимог та потреб.
- Надання API, що дозволяє підприємствам інтегрувати систему з різними платформами сторонніх розробників, такими як Telegram, Slack, LinkedIn, Jooble, Work.ua, G Suite та інші.

Серед недоліків варто виокремити:

- Складна для освоєння людьми, які не мали досвіду користування HRM-системами.
- Процес створення рівнів доступу для робітників виглядає заплутаним.
- Багато функціоналу недопрацьовані до кінця і не відчуваються закінченими.
- Деякі окремі елементи мають невеликі проблеми із продуктивністю.

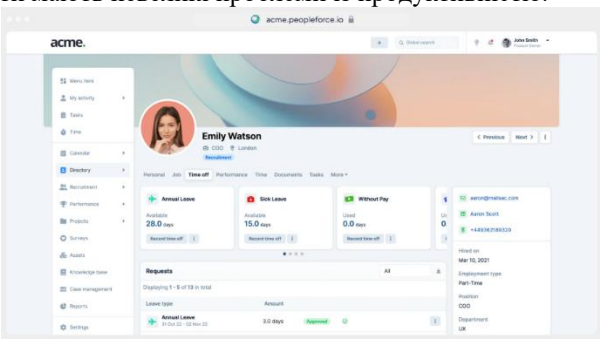


Рис. 2. Інтерфейс системи PeopleForce

HURMA [3] представляє собою систему управління людськими ресурсами (HRIS/HRM), спрямовану на раціоналізацію фінансових витрат та підвищення продуктивності (рис. 3). Головною метою є створення доступного інтегрованого рішення, спроможного оптимізувати процеси у сфері управління людськими ресурсами, найму та досягнення цілей управління для підприємств з чисельністю персоналу від 10 осіб.

До переваг слід віднести:

- Зручний інтерфейс.
- Гарно зроблене управління профілями співробітників.
- Управління відсутністю зроблене на високому рівні.
- Гарний збір та представлення статистики, розрахунок зарплати.
- Аналітика даних реалізована не доброму рівні.
- Відмінна робота служби підтримки, регулярні покращення і оновлення системи, виправлення помилок не займає багато часу.

До недоліків відносимо:

- Стає важче працювати якщо в компанії багато рекрутерів.
- Незручний процес перерозподілу прав доступу.
- Відсутність деяких функцій, що присутні у більшості розвинених HRMS.
- Немає вбудованого керування підпискою та обробки платежів.

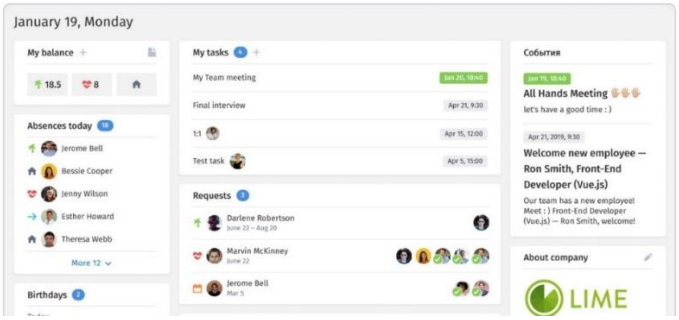


Рис. 3. Інтерфейс системи Hurma

У таблиці 1 наведемо узагальнене порівняння аналогів за певними характеристиками.

Таблиця 1.

Порівняння аналогічних розробок

Характеристика	HiBob	PeopleForce	Hurma	Власна розробка
Сумісність із платформами	Вебплатформа, мобільний застосунок	Вебплатформа	Вебплатформа	Вебплатформа
Загальна кількість функціоналу	5	5	4	3
Зручність інтерфейсу	5	4	4	4
Онбординг/ офбординг	5	3 офбординг відсутній	3 офбординг відсутній	2
Управління продуктивністю	4	4	3	0
Керування робітниками	5	4	4	4
Відслідковування робочого часу	5	5	4	4
Робота із документами	4	1	1	0
Робота із статистикою та аналітикою	5	5	5	3
Управління відсутністю	5	5	5	4

Якщо провести висновок аналізу аналогів, то можна побачити що HiBob, як найбільша із наведених програм-аналогів, очікувано випереджає інші наведені аналоги в багатьох аспектах. Але врешті кожна система управління персоналом має свої плюси і мінуси. Тому можна із впевненістю сказати що вибір системи управління персоналом залежить від індивідуальних потреб організації-користувача.

У випадку власної розробки для дипломного проекту не вистачає достатньої кількості ресурсів для навіть приблизної реалізації тих функцій, які було виявлено у подібних розробках. Адже більшість таких систем, мають величезну кількість функціоналу, які можна виокремити як великі окремі програми.

Було прийнято рішення створити систему, яка наблизиться до проаналізованих розробок, та створить усі умови для подальшого розвитку вебзастосунку. Власний продукт буде має містити деякі елементи аналогічних розробок.

Список використаних джерел

1 HiBob: Award-winning HR platform for all of your HRIS needs. HiBob. URL: https://www.hibob.com/

2. All-in-one HR платформа PeopleForce. All-in-One HR platform PeopleForce. URL: https://peopleforce.io/uk.

3. ATS system and HRM platform in one solution. HURMA. URL: https://hurma.work/.

УДК 004.7

Соловей О.І.
Антонюк Д.С., доцент кафедри інженерії програмного забезпечення, канд. пед. наук, доцент.
Державний університет «Житомирська політехніка»

Огляд проблематики переводу доменної структури Active Directory у хмару

В сучасному світі використання комп'ютерної техніки значно покращило ефективність роботи підприємств та організацій які виробляють безліч товарів та послуг. Поширеність техніки набула таких масштабів, що централізоване управління робочими місцями які обладнані ПК почало вимагати окремих рішень, одним з них яких стала доменна структура від компанії Microsoft. Це рішення пройшло довготривалий процес еволюції від домену на основі Windows NT до служби каталогів Active Directory і в теперішні дні еволюціонувало до хмарного рішення Microsoft Entra ID. Ця робота допоможе визначити певні особливості, проблематику і можливість застосування даних технологій як в одиночному так і гібридному режимі.

Відомо, що зміна інфраструктури це тривалий процес який вимагає чіткого планування можливих ризиків, обрахування бюджету та визначення обмежень з якими можна стикнутися як під час переходу із однієї структури до іншої так і при початковому розгортанні. Станом на 2020 рік за повідомленням Frost & Sullivan близько 90% компаній із списку Global Fortune 1000 використовували Microsoft's Active Directory (AD)[1].

Із початком ери хмарних технологій можливостей Active Directory стало не вистачати через брак інструментів інтеграції і до уваги світової спільноти, яка використовує комп'ютери в корпоративному секторі, було представлене хмарне рішення Microsoft Azure Active Directory (AAD) на сьогоднішній час відоме як Microsoft Entra ID і яке являє собою хмарну службу для керування користувачами та доступом до ресурсів. Розглянемо кожне із рішень яке застосовується сьогодні.

Active Directory – це служба каталогів від компанії Microsoft для доменних мереж Windows[2]. Ця служба полегшує керування мережевими ресурсами та ідентифікаторами користувачів у середовищі Windows. Спрощена схема Active Directory наведена на рис.1. В основі автентифікації лежать протоколи NTLM та Kerberos, а для виявлення ресурсів – LDAP. Основний функціонал AD включає:

- Автентифікація
- Авторизація
- Служба каталогів
- Управління груповою політикою

Рис. 1. Спрощена схема Active Directory

Entra ID – це хмарна реалізація служби каталогів та керування ідентифікацією та доступом яку надає Microsoft[3]. Попередня назва Azure Active Directory могла викликати певні асоціації із Active Directory, проте її функціонал орієнтований на підтримку web-служб, які використовують в своїй роботі RESTful інтерфейс (Office365, Google Apps, тощо) та надає деякі функціональні можливості локальної Active Directory у хмарі Azure[4]. У своїй роботі для автентифікації використовує протоколи SAML та OAuth 2.0 зокрема. Спрощена схема Entra ID наведена на рис.2. Entra ID пропонує низку функцій орієнтованих на захист хмарних програм, забезпечення відповідності і оптимізації IT-процесів, наприклад:

- Хмарне керування ідентифікацією
- Єдиний вхід (Single Sign-On) та багатофакторна автентифікація (MFA)
- Інтеграція додатків
- Сценарії ідентичності B2B та B2C

Рис. 2. Спрощена схема Entra ID

У Таблиці 1 наведено загальне порівняння подібностей та відмінностей між AD та Entra ID

Таблиця 1.

Порівняння Active Directory та Entra ID

	Active Directory	Entra ID
Керування користувачами та групами	Так	Так
Автентифікація	Так	Так
Авторизація	Так	Так
Централізоване керування	Так	Так
Покращена безпека	Так	Так
Оптимізований користувацький досвід	Так	Так
Протоколи	NTLM, Kerberos, LDAP	SAML, OAuth 2.0, OpenID Connect.
Групова політика	так	Замінено на Conditional Access policy
Служби домену	DNS, DHCP, NPS	ні
Керування пристроями користувачів	Локальні ресурси або ресурси підключені до локальної мережі	пристрої які отримують доступ до хмарних ресурсів, включаючи мобільні пристрої
B2B і B2C	ні	Доступ для зовнішніх партнерів і клієнтських програм
Інтеграція програм	Інтеграція з локальними ресурсами	Інтеграція з хмарними рішеннями

Тепер, коли відомі деякі подібності та відмінності у використанні цих двох технологій, можна визначити пріоритетне використанням одного чи іншого підходу. Даний вибір сильно обмежить можливості компанії і нівелює напрацьовані результати клієнтської бази, а також обмежити доступ до сучасних технологій або змусити здійснювати вартісні переноси інтегрованих програм у хмару. [5]

В такому випадку доцільним є використання гібридної інфраструктури яка поєднує в собі як використання on-prem систем в локальній мережі або розгортання контролеру домену в хмарі у якості віртуальної машини так і хмарних рішень які надаються компанією Microsoft. Ця технологія допомагає синхронізувати дані розміщенні на локальних серверах Active Directory та у хмарному сервісі Entra ID та отримати переваги обох технологій. Схема гібридної інфраструктури наведено у рис. 3.

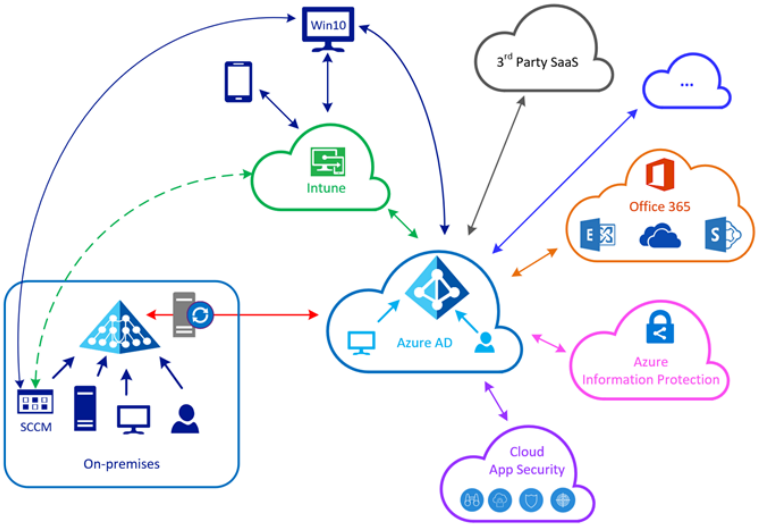


Рис. 3. Схема гібридної інфраструктури

Розглянемо деякі переваги використання гібридної інфраструктури:

- Полегшує інтеграцію існуючих бізнесів до хмарних технологій залишаючи при цьому можливість використання програм які були випущення для інтеграції із локальної інфраструктурою і використовують старіші протоколи автентифікації, зокрема ERP-системи. Зазвичай такі програмні продукти кастомізовані під певного користувача і перенос їх у хмару вимагає чимало часу і коштів
- Полегшить майбутній перехід до хмарних технологій, хоча цей шлях все ще виглядає досить складним
- Не потребує повторного розгортання структури користувачів в хмарі
- Полегшує керування пристроями підприємства і тому числі і мобільними пристроями – для цього потрібне лише підключення пристрою до мережі Internet
- Надає доступ до B2B та B2C технологій для підприємств.
- Дозволяє інтегрувати існуючих користувачів до хмарних продуктів чи сервісів таких як Office365 зокрема

Але поруч із перевагами присутні і певні недоліки:

- Оплата додаткових видатків на утримання та обслуговування хмари
- Необхідність вжиття заходів щодо убезпечення ще однієї можливої точки проникнення зловмисників
- Відсутність деревоподібної структури хмарного домену – збільшення навантаження на персонал для адміністрування декількох доменів (якщо в них є потреба)

Але якщо зважити всі недоліки та переваги - використання гібридної інфраструктури полегшує інтеграцію до новітніх сервісів та підходів які надають хмарні технології, відкривають нові шляхи для ведення бізнесу та здатні покращити співробітництво як із приватними так і корпоративними клієнтами за рахунок розширення аудиторії охоплення.

Список використаних джерел

1. Frost&Sullivan: Active Directory Holds the Keys to your Kingdom, but is it Secure? URL:https://www.frost.com/frost-perspectives/active-directory-holds-the-keys-to-your-kingdom-but-is-it-secure
2. Desmond, B., Richards, J., Allen, R., Lowe-Norris, A. (2013). Active Directory: Designing, Deploying, and Running Active Directory. Німеччина: O'Reilly Media, Incorporated.
3. Microsoft Azure Portal. URL: https://portal.azure.com/#view/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/~/Overview
4. Microsoft Azure Fundamentals: Define cloud models. URL: https://learn.microsoft.com/en-us/training/modules/describe-cloud-compute/5-define-cloud-models
5. Ed Price: Microsoft IT Part 1 - Hybrid Cloud Strategy. URL: https://techcommunity.microsoft.com/t5/azure-global/microsoft-it-part-1-hybrid-cloud-strategy/ba-p/306452

УДК 004.4

Новачук Р.О., аспірант
Антонюк Д.С., канд. пед. наук, доцент кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»

Огляд факторів що впливають на вибір та впровадження систем електронної комерції серед малих та середніх B2B підприємств

Перед вибором системи електронної комерції бізнесу необхідно уважно розглянути численні важливі фактори та потенційні обмеження. Важливо врахувати свої бізнес-цілі та вимоги потенційних клієнтів. Вдало обрана система можете забезпечити автоматизацію бізнес-процесів, знизити операційні витрати та вирішити широкий спектр задач, або ж, в протилежному випадку - створити нові проблеми або незручності. Різні платформи електронної комерції відрізняються не лише вартістю, але й наявністю різноманітних функцій, можливостями інтеграції, здатністю до модифікацій для відповідності до особливостей бізнесу, а також ціною подальшої технічної підтримки.

Для класифікації факторів, що вливають на вибір і впровадження систем електронної комерції можна використати technology organization environment (TOE) framework розроблений вченими Louis G. Tornatzky and Mitchell Fleischer ще у 1990 році для впровадження та реалізації технологічних інновацій [1]. Дана наукова праця є фундаментальною і використовується різними вченими в сучасних дослідженнях в області вивчення і впровадження електронних та інформаційних систем або технологій. Для прикладу, даний фреймворк був використаний науковцями для аналізу факторів що впливають на впровадження систем планування ресурсів (ERP) [2].

Згідно TOE фреймворку фактори що впливають на впровадження електронної системи діляться на три групи: технологічні фактори (Technological), організаційні фактори (Organizational), середовищні фактори (Environmental) (рис. 1).

У випадку технологічних факторів до цієї групи можна віднести:

- **Ціна і часовий ресурс** необхідний для імплементації та впровадження системи електронної комерції. В ціновій моделі варто звертати увагу не тільки на вартість створення, встановлення, впровадження, ліцензії, плати за місячною або річною підпискою, а також на цінову політику технічної підтримки та вартість подальших змін та видозмінення чи додавання нових модулів відповідних до потреб бізнесу [3].
- **Ключові функціональні елементи**. Кожна з існуючих систем електронної комерції надає певний набір функціональних частин по-замовчуванню або «з коробки». Детальний аналіз та оцінка таких частин на відповідність до вимог що ставить перед такою системою бізнес може на даному етапі виключити певні системи як такі, які не містять обов'язкових функціональних частин які є критичними для відповідності до бізнес процесів підприємства. Серед таких функцій можна виділити: ефективна система пошуку по каталогу, індивідуальний каталог для кожного з клієнтів, підтримка складних, налаштовуваних чи варіативних продуктів, підтримка динамічних та складних механізмів ціноутворення, системи квот та затвердження ордерів, підтримка системи ролей та прав, що відображається на бізнес процесах здійснення замовлень за B2B моделлю, можливість здійснення великих покупок з сотнями або тисячами позицій в замовленні, інтеграція з платіжними системами, поштовими сервісами, механізми автоматизованих регулярних замовлень, можливості моніторингу та управлення запасами на різних складах в режимі реального часу та багато іншого.
- **Інфраструктура та модульна архітектура**. Вендори програмного забезпечення пропонують різні моделі хостингу своїх рішень електронної комерції. Для прикладу SaaS рішення (software as a service) з використанням хмарних технологій Microsoft Azure, AWS, або Google Cloud забезпечують можливість швидкого автоматичного масштабування систем, що забезпечує стабільність функціонування в моменти підвищеного навантаження, а також можливість регулярного оновлення системи, що дає можливість отримувати оновлення та виправлення помилок, що, в свою чергу, значно продовжує термін експлуатації таких систем. В свою чергу модульна архітектура дозволяє розширювати функціональні можливості системи шляхом написання модулів, або доповнень, не змінюючи вихідний код системи.
- **Можливість інтеграції зі сторонніми системами**. Даний фактор часто є критичним для успішної автоматизації багатьох процесів. Необхідність і відповідна архітектурна і функціональна можливість інтеграції з такими системами як customer relationship management (CRM), enterprise resource planning (ERP), content management system (CMS), Product Information Management (PIM), маркетинговими та іншими системами – може бути одним з основних і вирішальних факторів у виборі на користь вендора, що пропонує рішення електронної комерції.

Рис. 1. Фактори вибору системи електронної комерції за TOE фреймворком

До групи організаційних факторів відносять:

- **Ресурси**. Достатня кількість фінансових, людських та інших ресурсів для впровадження та підтримки системи електронної комерції.
- **Навчання та підтримка**. Забезпечення навчання для працівників та наявність технічної підтримки для ефективного використання нової системи.
- **Розмір організації.** Великі компанії як правило мають більше ресурсів для впровадження нових систем, тому великі компанії мають більше можливостей для вибору.
- **Підтримка зі сторони менеджменту**. Ступінь розуміння та підтримка вищим керівництвом впровадження системи електронної комерції.

Група середовищних факторів складається з:

- **Конкуренція.** Бізнес може зазнавати тиску з боку конкурентів, що вже використовують системи електронної комерції або використовують системи, що мають конкурентну перевагу за своїми властивостями.
- **Постачальники та партнери.** Готовність та можливість постачальників та партнерів використовувати та інтегруватися з системою електронної комерції. Цей фактор особливо актуальний серед бізнесів які використовують B2B модель електронної комерції, так як в такій моделі часто є потреба автоматичного та регулярного здійснення транзакцій між системами покупців і продавця, що в свою чергу вимагає наявність програмних API для реалізації автоматичних процесів.
- **Клієнтські очікування**: Очікування клієнтів щодо доступності та функціональності системи електронної комерції. В даному випадку потенційні або існуючі клієнти можуть приймати рішення на користь одного або іншого постачальника залежно від наявних функціональних частин характерних системам електронної комерції.

Таким чином врахування факторів з трьох груп дає можливість зробити всебічний аналіз відповідності систем електронної комерції до потреб бізнесу, що в подальшому буде мати безпосередній ефект на ефективність її застосування та використання.

Список використаних джерел

1. Tornatzky, Louis G.; Fleischer, Mitchell (1990). The Processes of Technological Innovation. Issues in organization and management series. Lexington, Massachusetts: Lexington Books
2. Morawiec, P.; Sołtysik-Piorunkiewicz (2023). A. ERP System Development for Business Agility in Industry 4.0—A Literature Review Based on the TOE Framework. Sustainability 2023, 15, 4646. https://doi.org/10.3390/su15054646
3. 6 essential criteria to consider when selecting a B2B commerce vendor. URL: https://commercetools.com/blog/6-essential-criteria-to-consider-when-selecting-a-b2b-commerce-vendor.

УДК 004.4

Рудівський О.О.
Антонюк Д.С., к. пед. н., доцент, доцент кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»

Метод оцінки та забезпеченні якості архітектури програмного забезпечення на основі статичного та історичного аналізу коду

Якість архітектури програмного забезпечення є одним з визначальних чинників, що впливають на такі важливі аспекти, як швидкість розробки, кількість дефектів та передбачуваність процесу розробки і тестування в цілому. Архітектурний борг, як складова частина більш широкого поняття технічного боргу, має великий вплив також на здатність програмного забезпечення до адаптації щодо зміни вимог користувачів системи до функціональності, а також до зовнішніх умов, таких як кількість одночасних користувачів, зміни в апаратному забезпеченні чи доступність мережевих ресурсів. Високий рівень архітектурного боргу призводить до збільшення кількості збоїв та зниження задоволення користувачів. Це підкреслює важливість мінімізації архітектурного боргу та утримування його на цьому рівні [1].

Однією з ключових складових архітектури є топологія коду, яка визначає організацію коду в окремі модулі з чітко визначеними інтерфейсами. Метою такого розподілу є ізоляція пов’язаного коду всередині модулю і приховування його внутрішніх деталей від зовнішнього світу. Це дозволяє зменшити кількість модулів, які змінюються в процесі оновлення. Скорочення кількості змінених модулів, в свою чергу дозволяє пришвидшити процеси компіляції, автоматизованого тестування і розгортання, а отже і вартості програмного забезпечення. Розбиття програмної системи на модулі має і організаційні переваги, адже робота над різними модулями може відбуватися паралельно з мінімальними витратами на координацію між розробниками чи командами. Також, це спрощує як розуміння системи в цілому так і окремих модулів окремо один від одного.

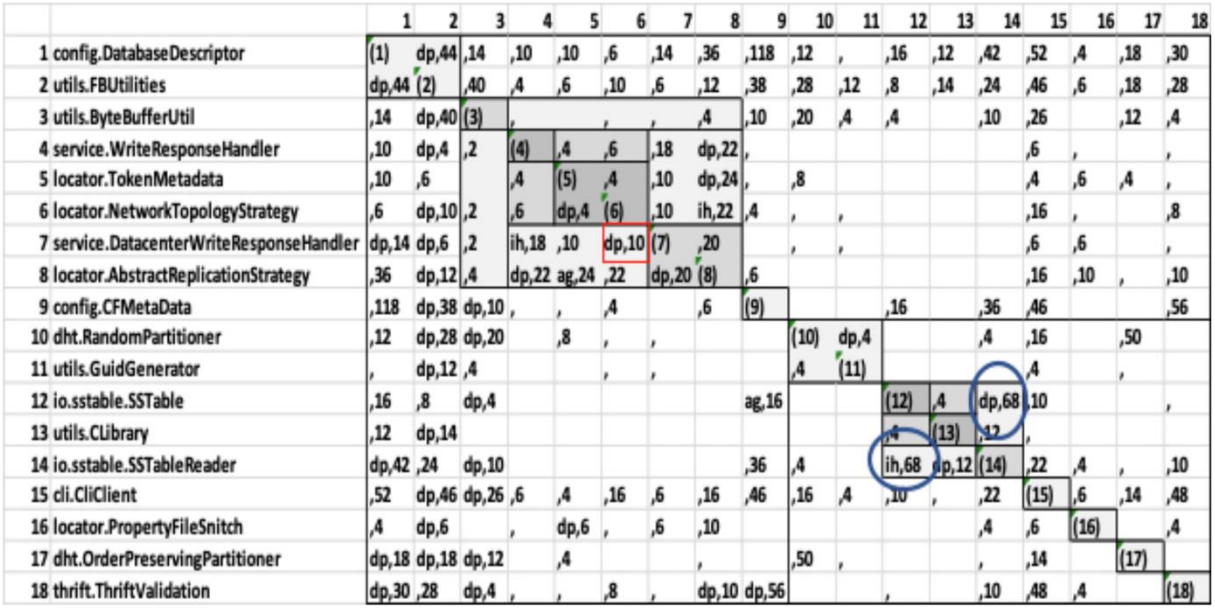
Ознакою якісної структури модулів є висока когезія (пов’язаність) всередині модулів та низька зв'язність між модулями. Когезія визначає, наскільки сильно пов’язані елементи модуля між собою. Високий рівень когезії означає, що модуль виконує одну функцію, а отже всі зміни в ньому будуть підпорядковані одній меті. Зв’язність визначає, наскільки сильно різні модулі пов’язані між собою. Низька зв’язність між модулями означає, що зміни в одному модулі зазвичай не вимагають змін в багатьох інших модулях.

Існує багато метрик, які дозволяють кількісно оцінити когезію та зв’язність модулів шляхом статичного аналізу коду. Проте цього недостатньо для повного розуміння якості архітектури, оскільки існують типи неявної зв'язності між модулями, які важко або навіть неможливо виявити за допомогою статичного аналізу коду.

Головними причинами виникнення неявної зв’язності є залежність від певної зовнішньої системи, формату файлів чи протоколу. Найбільш складний для виявлення тип зв’язності виникає коли модулі побудовані на основі знання про внутрішню будову інших модулів. У цьому випадку внутрішні зміни в одному з модулів автоматично призводять до необхідності внесення змін у всі залежні модулі. Ситуація ускладняється тим, що необхідність значної частини цих змін, не відома на початку процесу, що, в свою чергу, призводить до збільшення часу розробки та кількості дефектів які виникають в наслідок змін.

Одним з вирішень проблеми оцінки зв’язності між модулями полягає у поєднанні методів статичного та історичного аналізу коду. Аналіз історії змін файлів дозволяє виявити файли що змінюються одночасно [2]. Статичний аналіз коду дозволяє виявити всі файли які пов’язані між собою. Комбінація цих технік дозволяє виокремити підмножину взаємозалежних файлів яку неможливо або дуже важко виявити іншим чином.

Першим кроком є аналіз інформації з системи контролю версій. Це передбачає побудову так званої Design Structure Matrix (DSM) [3]. DSM являє собою матрицю суміжності, де елементами є кількість разів, коли файли, позначені відповідним рядком і стовпцем, змінювалися одночасно. Додатково в кожній комірці корисно додати інформацію про структурний зв’язок між файлами. Наприклад інформацію про виклик одного файлу з іншого або що файл наслідує інший файл.



		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	config.DatabaseDescriptor	(1)	dp,44	,14	,10	,10	,6	,14	,36	,118	,12	,	,16	,12	,42	,52	,4	,18	,30
2	utils.FBUtilities	dp,44	(2)	,40	,4	,6	,10	,6	,12	,38	,28	,12	,8	,14	,24	,46	,6	,18	,28
3	utils.ByteBufferUtil	,14	dp,40	(3)	,		,	,	,4	,10	,20	,4	,4		,10	,26		,12	,4
4	service.WriteResponseHandler	,10	dp,4	,2	(4)	,4	,6	,18	dp,22	,						,6	,		,
5	locator.TokenMetadata	,10	,6		,4	(5)	,4	,10	dp,24	,	,8					,4	,6	,4	,
6	locator.NetworkTopologyStrategy	,6	dp,10	,2	,6	dp,4	(6)	,10	ih,22	,4	,	,				,16	,		,8
7	service.DatacenterWriteResponseHandler	dp,14	dp,6	,2	ih,18	,10	dp,10	(7)	,20		,	,				,6	,6		,
8	locator.AbstractReplicationStrategy	,36	dp,12	,4	dp,22	ag,24	,22	dp,20	(8)	,6						,16	,10	,	,10
9	config.CFMetaData	,118	dp,38	dp,10	,	,	,4		,6	(9)			,16		,36	,46			,56
10	dht.RandomPartitioner	,12	dp,28	dp,20		,8	,	,			(10)	dp,4			,4	,16		,50	
11	utils.GuidGenerator	,	dp,12	,4			,	,			,4	(11)				,4		,	
12	io.sstable.SSTable	,16	,8	dp,4						ag,16			(12)	,4	dp,68	,10			,
13	utils.CLibrary	,12	dp,14										,4	(13)	,12	,			
14	io.sstable.SSTableReader	dp,42	,24	dp,10						,36	,4		ih,68	dp,12	(14)	,22	,4	,	,10
15	cli.CliClient	,52	dp,46	dp,26	,6	,4	,16	,6	,16	,46	,16	,4	,10	,	,22	(15)	,6	,14	,48
16	locator.PropertyFileSnitch	,4	dp,6		,	dp,6	,	,6	,10						,4	,6	(16)		,4
17	dht.OrderPreservingPartitioner	dp,18	dp,18	dp,12		,4			,		,50	,			,	,14		(17)	
18	thrift.ThriftValidation	dp,30	,28	dp,4	,	,	,8	,	dp,10	dp,56			,		,10	,48	,4		(18)

Рис. 1. Приклад DSM [2]

Файли, що часто або, навіть, завжди змінюються разом найімовірніше пов’язані між собою. Якщо такі файли належать до різних модулів, то це гарна ознака існування неявної зв’язності між модулями.

Останнім кроком аналізу є поєднання результатів історичного аналізу зі статичним аналізом коду. На цьому етапі знаходяться файли, які пов’язані між собою, проте не мають прямого зв’язку, який здатен виявити статичний аналіз коду.

Виявлені неявні зв'язки між модулями дозволяють здійснити структурні зміни та позбутися пов'язаного з ними архітектурного боргу. Зазвичай це передбачає об’єднання або розділення існуючих модулів, а також переміщення файлів між модулями. Результати цього аналізу дозволяють змінити структуру модулів так, що наступні оновлення вимагатимуть змін в меншій кількості модулів.

Перспективним є аналіз на основі часткових історій змін файлів з використанням техніки рухомого часового вікна. При цьому сам алгоритм аналізу залишається незмінним, проте звужується період історичного аналізу. Потім процес повторюється з початку з частковим зсувом історичного періоду. Можуть бути цікавими дві стратегії керування вікном аналізу. Перша це аналіз фіксованої кількості змін на кожному кроці. Друга можлива стратегія це аналіз усіх змін за певний проміжок часу. Вибір конкретної стратегії залежить від того, як саме відбувається процес розробки чи оновлень програмного забезпечення.

Аналіз на основі часткових історій змін файлів дозволить виявляти не лише характер зв'язності між модулями, але й те, як вона змінюється з часом. Це, в свою чергу, дає можливість зрозуміти не лише які модулі мають неявну зв'язність, а й у яких типах змін ця зв'язність проявляється, що відкриває можливості для подальшого вдосконалення структури модулів програмного продукту. Це дозволить відрізнити групи файлів які час від часу міняються синхронно, але переважна більшість змін в них незалежні, від файлів які переважно змінюються разом, і майже ніколи не змінюються окремо. Друга група файлів є значно кращим кандидатом для потенційної реорганізації модулів.

Список використаних джерел

1. M. Richards and N. Ford, Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020.

2. L. Bass, P. Clements, R. Kazman, Software Architecture in Practice (SEI Series in Software Engineering) 4th Edition. Addison-Wesley Professional, 2022

3. S. D. Eppinger, T. R. Browning, Design Structure Matrix Methods and Applications. The MIT Press, 2016

УДК 004.94

Москалик Д.О., аспірант
Антонюк Д.С., к. пед. н., доцент кафедри ІПЗ
Державний університет «Житомирська політехніка»

Підхід до моделювання процесу розробки програмного забезпечення для дослідження впливу архітектури на продуктивність розробки

На етапі планування нової програмної системи важливим кроком є вибір архітектурної моделі, яка має не тільки задовольняти всім функціональним вимогам та критеріям якості, а й сприяти продуктивній розробці такої системи та її ефективній подальшій підтримці. В сучасному світі ефективність процесу розробки рідко враховується при виборі архітектури через складність її оцінки, особливо в контексті різних архітектурних підходів. Одним із способів дослідження даної проблеми є моделювання процесу розробки програмного забезпечення на достатньо високому рівні деталізації, щоб враховувати різні аспекти впливу архітектури на продуктивність процесу розробки.

До поширених підходів симуляційного моделювання процесів розробки програмного забезпечення відносяться агентно-орієнтоване, дискретно-подійне, стохастичне моделювання, системна динаміка, моделювання на основі процесів, а також різні гібридні методи, що поєднують елементи декількох підходів для отримання більш точної та детальної моделі. Аналіз останніх наукових досліджень щодо моделювання процесу розробки програмного забезпечення [1] показав, що найбільш поширеними серед науковців є агентно-орієнтоване та дискретно-подійне моделювання, при чому популярність агентно-орієнтованого підходу з часом зростає, а застосування системної динаміки, навпаки, йде на спад. Системна динаміка зазвичай застосовується для високорівневого моделювання різних етапів розробки програмних проєктів та надає лише середні значення визначених метрик [2], проте останнім часом спостерігається тенденція до застосування також симуляцій більш низького рівня, таких як агентно-орієнтоване, для моделювання тих самих високорівневих процесів [3]. Гібридні симуляції зустрічаються рідше та зазвичай поєднують системну динаміку або з агентно-орієнтованим, або з дискретно-подійним підходами для одночасного моделювання на різних рівнях абстракції [4; 5].

Так як для дослідження впливу різних архітектурних підходів на ефективність розробки програмного забезпечення необхідний високий рівень деталізації внутрішніх процесів, що відбуваються під час розробки, тестування, розгортання та подальшої підтримки програмної системи, доцільно за основу симуляційної моделі взяти агентно-орієнтований або дискретно-подійний підхід.

Перевагами агентно-орієнтованих симуляцій є можливість моделювання складної поведінки системи, включно з динамічною взаємодією між агентами, а також здатністю агентів адаптуватись до змін середовища. Основними недоліками даного підходу є його обчислювальна складність та важкість валідації моделі. Дискретно-подійні симуляції є навпаки обчислювально ефективними, але їм бракує гнучкості для моделювання складних динамічних систем з комплексними взаємодіями між їх складовими елементами. Так, при застосуванні агентно-орієнтованої симуляційної моделі є можливість помітити та дослідити певні взаємозв’язки між елементами складної системи, які ще не були виявлені, але, так як для детального аналізу поведінки системи необхідно виконувати багато циклів симуляцій, обчислювальна ефективність дискретно-подійного підходу також має свої переваги.

На основі викладеного матеріалу є доцільним подальше дослідження гібридного підходу симуляції процесу розробки програмного забезпечення з рівнем деталізації агентно-орієнтованого та обчислювальною ефективністю дискретно-подійного симуляційних підходів.

Список використаних джерел

1. García-García, J.A., Enríquez, J.G., Ruiz, M., Arevalo, C. and Jiménez-Ramírez, A. Software process simulation modeling: systematic literature review. Computer Standards & Interfaces. №70. 2020. p.103425. DOI: 10.1016/j.csi.2020.103425
2. Liu, B., Zhang, H., Dong, L., Wang, Z. and Li, S. Metrics for software process simulation modeling. arXiv preprint arXiv:2301.06390. 2023. DOI: 10.48550/arXiv.2301.06390
3. Saravanos, A. and Curinga, M.X. Simulating the Software Development Lifecycle: The Waterfall Model. Applied system innovation. №6(6). 2023. p.108. DOI: 10.3390/asi6060108
4. Li, Y., Zhang, H., Dong, L., Liu, B. and Ma, J. Constructing a hybrid software process simulation model in practice: An exemplar from industry. In Proceedings of the International Conference on Software and System Processes. 2020. pp. 135-144. DOI: 10.1145/3379177.3388906
5. Helal, M., Rabelo, L., Sepúlveda, J. and Jones, A. A methodology for integrating and synchronizing the system dynamics and discrete event simulation paradigms. In Proceedings of the 25th international conference of the system dynamics society. Vol. 3. No. 3. System Dynamic Society. 2007. pp. 1-24.

УДК 004.4

Візгалов О.Ю.
Антонюк Д.С., доцент кафедри інженерії програмного забезпечення,
канд. пед. наук, доцент
Державний університет «Житомирська політехніка»

Поточний стан розвитку Reinforcement Learning та перспективи його використання

В сучасному світі важко переоцінити значення і популярність технологій штучного інтелекту. Все більшою стає кількість сфер, де інтегруються або посилюють свій вплив різного роду навчені моделі, за допомогою технологій машинного навчання. Одним з найцікавіших і найперспективніших напрямків на сьогодні є навчання з підкріпленням або Reinforcement Learning (RL). В основі даного підходу до машинного навчання лежить принцип тренування агентів (дійових осіб) за механізмом винагороди і покарання у певному середовищі, де відбувається оцінка стану на кожному етапі прийняття рішень з множини можливих дій. Ціллю є отримання оптимальної стратегії, за допомогою якої буде можливо максимізувати результуючу винагороду.

До реальних прикладів застосування Reinforcement Learning можна віднести:

- Додатки для автомобілів з автопілотом, а саме: оптимізація траєкторії руху, динамічна побудова маршруту, реагування на зміну дорожньої ситуації та уникнення аварійних ситуацій. Такі задачі з використанням машинного навчання з підкріпленням вирішені у програмному забезпеченні компанії Amazon під назвою «AWS DeepRacer» [1].
- Додатки для керування персональними фінансами, інвестиціями та збереженнями. Тут Reinforcement Learning може допомогти побудувати оптимальну стратегію прийняття фінансових рішень в залежності від різних зовнішніх економічних чинників та власного матеріального становища. Цей підхід був використаний у власній розробці фінансового симулятора [2]. Також існують рішення від компанії IBM [3], які дозволяють оптимізувати процес керування інвестиціями на основі історичних даних зміни ціни певних акцій.
- Рішення для комп'ютерного аналізу та синтезу природної мови у сфері Natural language processing, частковим випадком чого також є можливість знаходити відповіді на запитання користувача, що вирішується за допомогою великих мовних моделей [4], таких як, до прикладу, ChatGPT [5].
- Рішення у сфері медицини, які можуть допомогти діагностувати захворювання на основі клінічних даних та результату опитування пацієнта. Також такі моделі мають змогу знайти оптимальний протокол лікування, базуючись на попередньому досвіді RL системи [6].
- Вирішення комплексних ігрових задач від шахів та Ґо до відео-ігор. Мабуть одним з найпопулярніших алгоритмів сьогодення для гри в шахи є AlphaZero, який використовує модель Deep Reinforcement Learning, що тренуючись сама з собою дозволяє досягнути значних надлюдських результатів, пропонуючи різні оптимальні та суб-оптимальні варіанти вирішення шахових задач. Також цей алгоритм може використовуватись не тільки для навчання гри в шахи, а й інших комплексних завдань [7].

Наведений вище список – далеко не вичерпний. Застосування RL систем можна також зустріти у маркетингових і рекламних рішеннях, в автоматизованих конвеєрних лініях виробництва, рекомендаційних системах електронної комерції або медіа, роботизованих механізмів в місцях підвищеної небезпеки для людей тощо.

Оскільки Reinforcement Learning все ще є досить новим напрямком у машинному навчанні, в ньому присутній значний простір для подальшого дослідження, вдосконалення та вирішення потенційних проблем. Основне випробування полягає в тому, що досить важливо ретельно підібрати коректний алгоритм обчислення оптимальної політики прийняття рішень в залежності від поставленої цілі та наявного середовища. На сьогодні існує значна кількість різних способів та методів тренування моделей за допомогою навчання з підкріпленням (Q-learning, Deep Q Network, Policy Gradient Methods тощо), що з одного боку дає гнучку можливість оцінити різні алгоритми і їх ефективність в залежності від проблеми, що вирішується, а з іншого боку створює ризик не отримати бажаного результати, спробувавши значну кількість різних підходів. Особливо потрібно брати до уваги той факт, що тренування моделі починається з випадкових рішень агента, які згодом піддаються оцінці – винагородами або покараннями, в залежності від фінального результату. Це все, в свою чергу, зумовлює потребу для такої моделі в значних обчислювальних потужностях і відчутних часових витрат. Тому якщо бажана модель не дасть результату, доведеться не тільки повторно реалізувати новий підхід, а й очікувати нових сесій тренувального процесу, що в залежності від умов і середовища може займати від декількох днів до декількох місяців.

На сьогоднішній день одним з найуспішніших алгоритмів для Reinforcement Learning є Soft Actor-Critic (SAC) [10], який зокрема використовується компанією OpenAI. Проте враховуючи значну складність, кількість параметрів, коефіцієнтів, розмірів буферу та залежність від рівнів середовища, інтеграція даного

алгоритму може стати справжнім технічним викликом. Зміна параметру може випадково вплинути на позитивний результат тренування, що не буде гарантувати стабільність системи. Тому для досягнення об'єктивних висновків потрібно бути готовим до запуску репрезентативної серії тренувань і впровадження багатошарових метрик вимірювання стабільності створеної системи.

Не дивлячись на виклики, які ставить перед нами вирішення задач за допомогою Reinforcement Learning систем, даний підхід показав себе як працююче і ефективне рішення для багатьох сфер нашого життя. Особливо актуальним даний напрям машинного навчання може бути у зв'язку зі збройною агресією російської федерації проти України, зважаючи на технологічність і цифровізацію сучасних методів ведення війни. Оскільки RL системи вже використовуються для автопілотів автомобілів, схожі принципи потенційно можуть бути реалізовані у безпілотних літальних апаратах, а також роботизованих механічних наземних системах, що говорить про надзвичайні перспективи даного напрямку.

Список використаних джерел

1. AWS DeppRacer. URL: https://aws.amazon.com/deepracer/
2. D.S. Antoniuk, T.A. Vakaliuk, V.V. Didkivskyi, O.Y. Vizghalov, Development of a simulator to determine personal financial strategies using machine learning, Ceur workshop proceedings, 2022.
3. Reinforcement Learning: The Business Use Case, Part 2. URL: https://medium.com/ibm-data-ai/reinforcement-learning-the-business-use-case-part-2-c175740999.
4. Aligning LLMs With Human Expectations: An Overview. URL: https://medium.com/@masoumzadeh/aligning-llms-with-human-expectations-an-overview-f2eb50e330d6
5. ChatGPT. URL: https://openai.com/chatgpt/
6. Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. 2021. Reinforcement learning in healthcare: A survey. ACM Computing Surveys (CSUR) 55, 1 (2021), 1--36.
7. AlphaZero: Shedding new light on chess, shogi, and Go. URL: https://deepmind.google/discover/blog/alphazero-shedding-new-light-on-chess-shogi-and-go/
8. Modern Reinforcement Learning approach and its applications. URL: https://sii.pl/blog/en/modern-reinforcement-learning-approach-and-its-applications/
9. Why You (Probably) Shouldn't Use Reinforcement Learning. URL: https://towardsdatascience.com/why-you-shouldnt-use-reinforcement-learning-163bae193da8
10. Soft Actor-Critic. URL: https://spinningup.openai.com/en/latest/algorithms/sac.html