DOI: https://doi.org/10.26642/ten-2025-1(95)-249-258
УДК 004.7:62-52

В.В. Воротніков, д.т.н., доц.
О.О. Шелуха, к.т.н.
К.І. Матвєєв, ст. викладач
Державний університет «Житомирська політехніка»

Система моніторингу користувацької активності та автоматизованого керування офісною IT-інфраструктурою

У статті представлено концепцію та реалізацію інтелектуальної мікросервісної платформи для моніторингу активності користувачів та автоматизованого керування офісною IT-інфраструктурою. Актуальність дослідження обґрунтовується стрімким ускладненням IT-середовищ, зростанням кіберзагроз і потребою у забезпеченні прозорості функціонування та інформаційної безпеки. Запропоноване рішення базується на подієво-орієнтованій архітектурі з використанням MQTT-шини як транспортного рівня, що забезпечує масштабованість, гнучкість і стійкість до збоїв. Система складається з трьох логічних рівнів: агентів збору даних, брокера повідомлень і аналітично-керуючих мікросервісів, що разом дозволяє виявляти аномалії, оптимізувати ресурси, а також реалізовувати автоматизовану реакцію на події в режимі реального часу.

У роботі детально описано функціональність програмного агента, розробленого на Python, який здійснює збір технічних і поведінкових параметрів системи, таких як завантаження процесора, активність користувача, мережевий стан тощо. Представлено архітектуру взаємодії між агентами, брокером та мікросервісами аналітики, що дозволяє проводити попередню фільтрацію, агрегацію та аналіз подій. Особлива увага приділена безпечній роботі агентів, використанню шифрування, кешуванню подій та захисту конфіденційності, збереженню даних у разі втрати зв'язку та технологіям гнучкого аналізу та візуалізації. Платформа орієнтована на інтеграцію з технологіями штучного інтелекту, машинного навчання та автоматизації, що відкриває перспективи для побудови адаптивних, самонавчальних систем управління офісною або промисловою IT-інфраструктурою, а запропонований підхід дозволяє перейти від класичних засобів моніторингу до створення гнучкої, адаптивної та інтелектуальної системи, орієнтованої на концепції Індустрії 4.0, де кожен персональний комп'ютер стає активним учасником цифрової екосистеми.

Ключові слова: *інтелектуальна система; моніторинг активності користувача; автоматизоване управління; мікросервісна архітектура; MQTT; автоматизація обслуговування; централізоване керування; офісна IT-інфраструктура; машинне навчання; аномалії в поведінці; агентський моніторинг; Індустрія 4.0.*

Актуальність теми. В сучасних умовах швидкого розвитку цифрового ландшафту, у зв'язку з широким впровадженням різноманітних технологій, зростанням кількості кіберзагроз різні технологічні компанії та промислові підприємства стикаються із необхідністю у все більш ефективному контролі IT-інфраструктури та моніторингу поведінки користувачів. Складність сучасних IT-систем та бізнес-процесів, що їх використовують, постійно зростає, це може призводити до зниження прозорості контролю, появи неочевидних «вузьких місць» та збільшення активності різноманітного прихованого шкідливого програмного забезпечення.

Традиційні підходи до моніторингу, які часто обмежуються базовим технічним контролем або вимагають ручного втручання, а також класичні методи виявлення процесів, що базуються на спрощених моделях (наприклад, Directly-Follows Graph) або припускають унікальність ідентифікаторів подій, виявляються недостатніми для забезпечення сучасного рівня інформаційної безпеки та ефективності використання ресурсів. Зокрема, стандартні методи виявлення процесів мають значні обмеження до застосування у складних випадках та не можуть ефективно працювати з багаторазовою появою одного типу події в межах одного процесу, що призводить до неточних або некоректних схем функціонування процесів та надмірної узагальненості.

Для вирішення зазначених проблем можуть використовуватися різноманітні засоби автоматизації, такі як нейронні мережі (зокрема, графові GNN та згорткові CNN). Ці системи використовують дані моніторингу активності користувачів (User Activity Monitoring – UAM) та журналів подій як основну інформаційну базу для аналізу. Робота з цими даними вимагає ефективних технік попередньої обробки, оскільки вони можуть містити шум, аномалії, пропущені події або некоректні формати.

Такі інтелектуальні системи із застосовуванням аналітичних підходів мають свої можливості:

1. Виявляти аномальну або підозрілу активність. Аналізуючи шаблони дій користувачів у журналах подій, системи можуть ідентифікувати викиди, «хаотичну» поведінку, що не відповідає типовому потоку процесу, аномалії та відхилення від встановлених правил процесу. Методи машинного навчання, такі як GNN та CNN, є потужними інструментами для такого аналізу, особливо у складних сценаріях та з зашумленими даними;

2. Реагувати в режимі реального часу та підтримувати автоматизоване керування. Концепція RPA (роботизована автоматизація процесів) та здатність систем на основі ШІ формувати автоматизовану підтримку прийняття рішень та прогнозувати кроки процесу вказують на потенціал для автоматизації дій на основі результатів моніторингу та аналізу. Виявлені моделі процесів можуть бути використані для їх подальшої автоматизації;

3. Оптимізувати використання енергії та апаратних ресурсів. Підвищення ефективності робочих процесів, виявлене завдяки моніторингу та аналізу, може опосередковано сприяти більш раціональному використанню загальних IT-ресурсів;

4. Забезпечувати прозорість у використанні IT-інфраструктури та виконанні процесів. Використання методів виявлення процесів для побудови схеми з журналів подій робить реальний потік робіт видимим та зрозумілим, підвищуючи прозорість складних процесів. Систематичне ведення та аналіз журналів активності користувачів (UAM) створює детальну інформаційну базу про використання систем та виконання дій, що також сприяє прозорості.

Важливою перевагою сучасних підходів є їх здатність долати обмеження традиційних методів щодо обробки складних структур та випадків з багаторазовою появою одного типу події, що досягається зокрема за рахунок використання багатоаспектного вбудовування та матриць схожості контекстів та дозволяє отримати більш точні схеми процесів порівняно зі стандартними інструментами.

Все зазначене підкреслює актуальність теми та можливість для розробки та впровадження інтелектуальних систем, які базуються на передових методах машинного навчання.

Аналіз останніх досліджень та публікацій. В роботі [1] автори пропонують метод виявлення процесів у логах, де можливе повторення одного типу події, що характерно для моніторингу активності користувачів в офісних IT-системах. Використано графові та згорткові нейромережі для точнішого виявлення шаблонів дій. Метод дозволяє моделювати складні офісні процеси з повторюваними діями працівників. До переваг зазначеного методу варто зарахувати те, що він дозволяє обробляти багаторазові події, типові для журналів користувацької активності та використовує GNN і CNN для високої точності виявлення патернів. Тим не менш застосування описаного методу має високу обчислювальну складність та потребує якісно маркованих даних для створення датасету, на базі якого буде виконуватися навчання моделей.

Автор роботи [2] розглядає моніторинг активності користувачів як ключовий інструмент у процесах роботизованої автоматизації (RPA). Так дані зазначеного моніторингу допомагають ідентифікувати рутинні завдання, перевіряти дотримання політик безпеки та створювати навчальні вибірки для моделей штучного інтелекту, що забезпечує інтелектуалізацію та автоматизацію тих чи інших офісних процесів. До переваг зазначеного методу можна зарахувати те, що він допомагає оптимізувати вибір задач для RPA, та виявлення аномалій, що потенційно дозволить покращити безпеку інформаційної системи компанії, хоча і може створювати певні ризики конфіденційності. Також варто зауважити, що якість результатів залежить також від точності зібраних даних.

У роботі [3] запропоновано концепцію Desktop Activity Mining для точного збору дій користувача на рівні інтерфейсу з метою побудови точніших моделей бізнес-процесів. Такий підхід дозволяє виявити реальні варіації виконання офісних задач і підвищити ефективність автоматизації. Також зазначений у роботі метод добре підходить для аналізу поведінки користувачів в офісних IT-середовищах. Серед переваг зазначеної концепції є те, що вона дозволяє більш деталізовано збирати дані щодо активностей користувачів, що своєю чергою дозволить підвищити точність побудови моделей для машинного навчання. Аналогічно до попереднього методу розглянута в роботі концепція має певні ризики щодо безпеки конфіденційних даних, оскільки вимагає встановлення на кожний ПК.

У досліджені [4] авторами запропоновано метод розпізнавання користувацьких дій за допомогою OCR та аналізу скріншотів, коли прямий доступ до логів неможливий. Підхід дозволяє реконструювати бізнес-процеси навіть у віртуалізованих або захищених середовищах та відкриває нові можливості для моніторингу IT-інфраструктури без втручання у системи, і зменшує ризик несанкціонованого доступу до чутливих даних. Проте зазначена система має недоліки швидкодії та може мати певні складності при зміні оформлення інтерфейсу.

Автори роботи [5] пропонують метод автоматичного генерування RPA-сценаріїв з логів користувача без попереднього моделювання. Зазначається, що запропонована система самостійно буде ідентифікувати окремі процеси в логах та створювати виконувані скрипти, що дозволить знизити вимоги до ручного аналізу й полегшить автоматизацію рутинних офісних задач. Проте така система матиме певну складність обробки складних сценаріїв з міжпроцесорними залежностями та потребує попередньої фільтрації великої кількості надмірної інформації у логах.

У [6] автори розглядають метод, що автоматично конвертує логи інтерфейсу користувачів у RPA-флоучарти, що може спростити створення сценаріїв автоматизації шляхом використання процесного майнінгу з урахуванням структури даних. Такий підхід може підвищити точність та зменшити час проєктування автоматизованих процесів в офісному IT-середовищі. Проте слід зауважити, що застосування такого методу вимагає уніфікованих логів з повними метаданими, і, відповідно, не всі стандартні алгоритми процесного майнінгу придатні для прямого переносу в RPA.

У статті [7] розглядається використання процесного майнінгу як інструменту для оцінки доцільності автоматизації конкретних процесів. Авторами пропонуються принципи відбору задач на основі частоти повторень, що особливо корисно для офісних IT-процесів, такий підхід орієнтований на стратегічне планування автоматизації, враховує реальні поведінкові патерни користувачів та надає можливість вибору, але в роботі не наведено жодних пропозицій щодо реалізації або архітектури запропонованої системи

У [8] авторами представлено систему, що дозволяє спостерігати за поведінкою користувачів в інтерфейсі та автоматично сформувати RPA-правила, що може зменшити потребу у ручному програмуванні сценаріїв. Такий підхід добре підходить для типових офісних задач, оскільки дозволяє повністю автоматизувати збір правил на основі поведінки, але може не враховувати складні сценарії та залежати від якості даних та дій користувача.

У роботі [9] авторами розглянуто застосування RPA для моніторингу інформаційних систем шляхом емуляції поведінки реального користувача, що дозволяє імітувати використання систем у реальному часі, фіксувати затримки, помилки та виявляти проблеми, які стандартні моніторингові засоби не виявляють. До недоліків такої системи можна зарахувати складність налаштування ботів та значне навантаження на систему при частому виконанні таких імітацій, але цей метод може бути ефективно адаптований до офісної IT-інфраструктури.

Авторами роботи [10] розглянуто централізоване управління великими фермами RPA-ботів, з акцентом на фінансову ефективність, безпеку, передачу знань та управління ресурсами. Така архітектура може бути корисною для масштабного моніторингу та автоматизованого керування офісною IT-інфраструктурою. Але в роботі також мало описано технічних деталей для реалізації даної системи.

Проведений аналіз підтверджує високу актуальність розробки систем, що зможуть інтегрувати моніторинг активності для виявлення аномалій або підозрілої поведінки системи та використовувати інтелектуальний аналіз даних з використанням методів машинного навчання для забезпечення більш комплексного та автоматизованого контролю IT-інфраструктури та поведінки користувачів.

Отже, розробка системи моніторингу користувацької активності та автоматизованого керування IT-інфраструктурою відповідає поточним викликам, що пов’язані зі зростанням складності IT-середовищ та потребою у підвищенні рівня безпеки, ефективності та прозорості їх функціонування.

Метою статті є розробка архітектури інтелектуальної мікросервісної платформи для моніторингу, управління та аналізу інженерних систем, на основі якої автори в подальшому створюють датасети для машинного навчання моделей на базі нейронних мереж для забезпечення автоматизованої роботи інтелектуальної системи моніторингу активності комп’ютерних систем з можливістю використовувати технології штучного інтелекту, машинного навчання та аналітики в комплексі, що є необхідним для створення підприємства Індустрії 4.0 [3].

Викладення основного матеріалу. Запропонована система моніторингу базується на подієво-орієнтованій мікросервісній архітектурі з використанням шини обміну повідомленнями MQTT. Вона забезпечує масштабованість, стійкість до збоїв та гнучкість у розгортанні і розвитку функціональності. Основу системи складають три логічні рівні: агенти збору даних, транспортний рівень (брокер), а також рівень обробки й взаємодії із зовнішніми сервісами. Така архітектура надає гнучкість, масштабованість, але вимагає системності в побудові представлення даних та правил роботи з ними. Структура представлена на рисунку 1.

Розроблена система ґрунтується на розподіленій архітектурі з використанням протоколу MQTT як центральної шини обміну повідомленнями між агентами, сервісами, аналітичними модулями, модулями машинного навчання та кінцевим інтерфейсом користувача.

Агентський рівень (Edge Layer)

Агент реалізується на мові програмування Python, версії 3.10 або вище, що забезпечує сумісність із сучасними бібліотеками та підтримку асинхронного виконання. Вибір Python зумовлений його простотою, широкою екосистемою інструментів для роботи з системними ресурсами, мережею та паралельним програмуванням. Для збору системної інформації застосовується бібліотека psutil, що дозволяє отримувати показники завантаження процесора, використання оперативної пам’яті, інформацію про диски, мережеві інтерфейси та запущені процеси. Вона забезпечує кросплатформну взаємодію з операційною системою на низькому рівні.

Моніторинг активності користувача, зокрема відстеження натискань клавіш і руху миші, реалізується за допомогою бібліотек pynput (для Windows та Linux) або pyxhook (зокрема для X11 у Linux). Це дає можливість фіксувати взаємодію користувача із системою без втручання в її роботу. Для зв’язку з

брокером повідомлень MQTT використовується бібліотека paho-mqtt, яка є офіційним клієнтом від Eclipse. Вона дозволяє здійснювати публікацію даних, підписку на теми, а також обробку втрати підключення та повторного з'єднання. Паралельне виконання різних служб (моніторів, планувальника, публікатора тощо) досягається за допомогою модулів threading або asyncio – залежно від вибраного стилю програмування. Обидва підходи забезпечують ефективне виконання незалежних задач, що працюють у фоновому режимі.

Для тимчасового збереження даних або черги подій використовуються модулі json (людинозчитуваний формат) та pickle (бінарне серіалізоване представлення). Вони дозволяють зберігати та відновлювати події з диску у випадку втрати мережі. Інструменти на кшталт watchdog використовуються для реагування на зміни у файловій системі, platform – для визначення ОС, а uuid – для створення унікального ідентифікатора пристрою. Це важливо для ідентифікації джерела даних при роботі з кількома агентами у мережі.

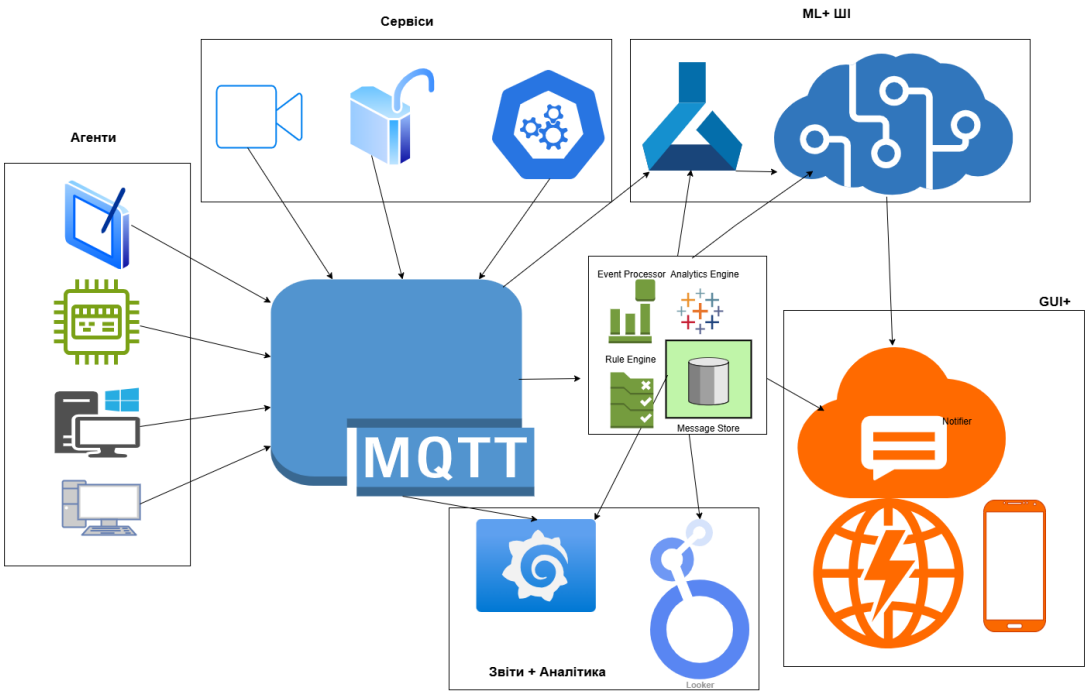

Рис. 1. Структура архітектури пропонованого рішення

Загалом набір технологій орієнтований на мінімальну залежність від стороннього ПЗ, простоту встановлення, кросплатформенність та надійність у роботі з системами автоматизації, моніторингу та телеметрії. На периферії системи функціонують програмні агенти, встановлені на кінцевих пристроях – персональних комп'ютерах, контролерах, сенсорних вузлах тощо. Їхнім основним завданням є безперервне збирання телеметричних даних, таких як навантаження на системні ресурси, параметри навколишнього середовища, активність користувача або показники роботи обладнання. Агенти є повноцінними учасниками MQTT-шини, виконуючи роль публікаторів (publishers) відповідних тем (topics). Крім агентів можна використати інформацію з інших систем.

На клієнтських пристроях функціонують агенти моніторингу, реалізовані як фонові Python-процеси або окремі служби. Кожен агент виконує збір технічних та поведінкових параметрів, серед яких:

- завантаження центрального процесора та оперативної пам'яті;
- активність користувача (рухи миші, натискання клавіш);
- наявність мережевого з'єднання;
- робота програм, процесів, служб.

Агент функціонує незалежно від підключення до центрального сервера: у разі втрати зв'язку, критичні події кешуються локально і надсилаються повторно після відновлення з'єднання. Передача інформації відбувається у вигляді MQTT-повідомлень із визначеним форматом тем, що дозволяє ефективно маршрутизувати події до відповідних обробників. Запропонована архітектура представлена на рисунку 2. Архітектура агента дозволяє додавати нові джерела подій (наприклад, API моніторингу GPU, температури, підключення USB). Фільтри події на боці клієнта дозволяють мінімізувати об'єм інформації та провести первинну обробку даних, а шифрування даних дозволяє вберегти чутливі дані (за потреби). Механізми публікації через MQTT з TLS, підпис повідомлень (опціонально), мінімальний набір прав – лише читання системної інформації дозволяє не перетворювати агента на джерело витоку інформації.

Агент складається з кількох модулів, що функціонують як окремі потоки або асинхронні задачі:

- конфігурація – завантаження налаштувань агента з конфігураційного файлу або сервера;
- менеджер потоків – створює та керує виконанням окремих служб у вигляді потоків або асинхронних задач;
- MQTT-клієнт – встановлює з'єднання з MQTT-брокером для обміну повідомленнями.

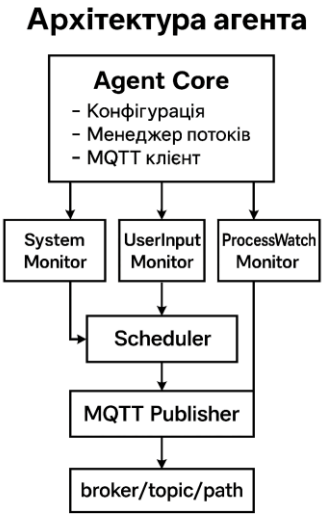

Рис. 2. Структура архітектури пропонованого агента

System Monitor

Служба моніторингу системних ресурсів (рис. 3):

- отримує інформацію про навантаження процесора, пам'яті, використання дисків і мережі;
- застосовує бібліотеку psutil для збору даних;
- публікує події у Scheduler при виявленні аномалій або перевищень порогів.

Рис. 3. Фрагмент метрик, представлених у брокері

UserInput Monitor

Служба відстеження дій користувача (рис. 4):

- використовує глобальні хуки клавіатури та миші через бібліотеки pynput або keyboard;
- фіксує активність, без зберігання змісту дій (з метою приватності);
- дозволяє оцінити рівень взаємодії користувача із системою.

```
Key pressed: 'f'
Mouse moved to (992, 660)
Mouse moved to (992, 661)
Mouse moved to (992, 662)
Key pressed: Key.ctrl_l
Key pressed: '\x03'
Key pressed: Key.cmd
Key pressed: Key.shift
Key pressed: 'S'
Mouse moved to (992, 663)
```

Рис. 4. Фрагмент метрик, отримаших з pynput

ProcessWatch Monitor

Служба контролю за запущеними процесами дозволяє відстежувати створення або завершення процесів, може мати список важливих або заборонених процесів та корисна для безпеки і моніторингу продуктивності. Процеси збираються та групуються, що дозволяє визначати відсоток завантаженості та процес, який споживає ресурси. Такий підхід дозволяє визначити шкідливе ПЗ. Приклад отримання інформації представлений у фрагменті нижче:

Process dwm.exe (PID: 1344)
CPU: 0.0%
Memory: 52.36 MB
Path: C:\Windows\System32\dwm.exe

Process svchost.exe (PID: 1472)
CPU: 0.0%
Memory: 6.44 MB
Path: C:\Windows\System32\svchost.exe

Пошук шкідливого ПЗ можна виконувати статистикою завантаженості, реакцією на виклик диспетчера задач та сортуванням процесів за шляхами, які не знаходяться у відомих директоріях (Program Files, System32, Windows), особливо звертаючи увагу на підозрілі (AppData\Local\Temp, AppData\Roaming, Documents або Downloads), також можна порівнювати шляхи з білим списком довірених програм.

Scheduler

Планувальник подій – це центральний логічний блок, що відповідає за обробку подій, отриманих від різних моніторингових модулів агента. Його головна функція полягає у тому, щоб забезпечити впорядкований і ефективний потік інформації до модуля публікації MQTT, а також зменшити навантаження на мережу та уникнути надмірного дублювання даних.

Після отримання події від монітора Scheduler спершу перевіряє її на актуальність та унікальність. Він може фільтрувати зайві або повторювані події, що не містять нової інформації, а також дотримуватися заданої логіки пріоритетів і виключень. Наприклад, якщо процесор стабільно працює в межах допустимого навантаження, подія може бути пропущена або затримана до моменту, коли буде зафіксовано суттєве відхилення.

Крім фільтрації, Scheduler також займається агрегацією подій. Це означає, що за певний проміжок часу він може зібрати кілька подій одного типу, узагальнити їх і передати вже у вигляді одного зведеного повідомлення. Такий підхід дозволяє значно зменшити кількість мережевих запитів, об’єднуючи дані про навантаження системи, активність користувача чи роботу процесів у стислий формат.

У межах обробки подій Scheduler може обчислювати додаткові параметри, наприклад, зміну значення у часі або відповідність певним пороговим умовам. Після завершення обробки він формує структуровану подію у вигляді стандартного JSON-об’єкта з єдиною схемою. До повідомлення обов’язково додаються такі елементи, як час події, ідентифікатор агента, тип події, джерело і зміст події (payload), що може містити числові значення, статуси або повідомлення.

Сформована подія передається до модуля публікації MQTT. Якщо мережеве з’єднання недоступне, Scheduler може скористатися чергою подій, яка тимчасово зберігає інформацію до моменту відновлення зв’язку. Таким чином, планувальник не лише впорядковує потік подій, а й є гарантом їхньої цілісності та ефективності обробки.

Scheduler може працювати в режимі реального часу, коли події обробляються негайно, або в режимі квазіреального часу, коли обробка відбувається пакетно через задані інтервали. Гібридний підхід дозволяє критичні події передавати одразу, а менш важливі – накопичувати та обробляти з певною затримкою. Архітектура цього модуля передбачає можливість масштабування та розширення, наприклад, через динамічну зміну правил фільтрації або інтеграцію з аналітичними модулями для виявлення аномалій.

MQTT Publisher

Публікатор MQTT Publisher – це служба, що відповідає за безпосереднє відправлення підготовлених повідомлень на MQTT-брокер. Її основне завдання полягає в тому, щоб забезпечити надійну та своєчасну передачу подій, сформованих планувальником Scheduler, до зовнішньої системи або хмари, де вони будуть далі оброблятися, зберігатися чи візуалізуватися.

Кожне повідомлення, що передається через MQTT Publisher, містить кілька ключових елементів. По-перше, це точна мітка часу (таймстемп), яка фіксує момент виникнення події. По-друге, вказується тип події – наприклад, cpu_usage, user_activity, process_state_change або інший, що допомагає класифікувати події на приймальному боці. По-третє, повідомлення включає самі дані – це може бути числове значення навантаження процесора, статус підключення, ім'я активного процесу або інші параметри, залежно від типу події.

MQTT Publisher працює асинхронно та підтримує механізм чергування. У випадках, коли підключення до брокера недоступне, служба взаємодіє з EventQueue – окремим модулем, що тимчасово зберігає події на диску або в локальній пам'яті. Після відновлення мережі або підключення до брокера публікатор автоматично надсилає накопичені повідомлення, зберігаючи послідовність і часову прив'язку.

Архітектурно MQTT Publisher може підтримувати різні рівні надійності доставки (QoS), налаштування збереження повідомлень (retain flag), а також іменування тем відповідно до заданої структури, наприклад, agent/{agent_id}/event/{event_type}. Такий підхід дозволяє масштабувати систему і чітко розділяти потоки даних між кількома агентами або модулями прийому.

У підсумку, MQTT Publisher є ключовим елементом взаємодії агента з зовнішнім середовищем, забезпечуючи передачу подій у стандартизованому форматі, з високою надійністю та гнучкістю у роботі з мережею.

Зразок тем MQTT

- agent/pc001/sysinfo/cpu → { usage: 37.5 }
- agent/pc001/sysinfo/mem → { used: 4096, total: 8192 }
- agent/pc001/user/activity → { mouse: True, keyboard: False }
- agent/pc001/sysinfo/processes → [{"name": "chrome", "cpu": 10.2}]
- agent/pc001/logs/offline → [...] (масив подій за час втрати зв'язку)

EventQueue, Logger

EventQueue та Logger – це дві допоміжні, але критично важливі служби агента, що забезпечують надійність, стійкість до збоїв та відстеження внутрішніх процесів. Вони дозволяють системі працювати навіть у нестабільних умовах, коли недоступне мережеве з'єднання або виникають внутрішні помилки.

EventQueue – це механізм черги подій, що активується у разі втрати зв'язку з MQTT-брокером. Якщо служба публікації не може надіслати повідомлення, події зберігаються тимчасово у локальній файловій системі або у вбудованій базі даних, такій як SQLite. Завдяки цьому жодна подія не буде втрачена. Система відстежує момент відновлення мережі і одразу ж починає надсилати накопичені повідомлення у правильному порядку, враховуючи хронологію та унікальність. Такий підхід гарантує цілісність даних навіть у складних умовах експлуатації, наприклад, на віддалених об'єктах з нестабільним інтернетом.

Logger своєю чергою відповідає за ведення журналу важливих подій, винятків, помилок, попереджень та інших внутрішніх станів агента. Цей компонент фіксує інформацію у зручному для аналізу форматі, часто із зазначенням часу, джерела події та типу повідомлення (наприклад, ERROR, WARNING, INFO). Журнали можуть зберігатися у текстових файлах локально, у системному журналі (наприклад, через syslog) або передаватися у зовнішню систему логування.

Логування особливо важливе для налагодження та аудиту. Воно дозволяє виявити, чому певна подія не була надіслана, коли зникла мережа, які дії виконував агент, і чи були вони успішними. Крім того, Logger може бути налаштований на різні рівні чутливості, а також інтегруватися з системами сповіщення про інциденти, якщо виявлено критичну ситуацію.

Разом ці служби створюють надійне середовище для роботи агента в реальному або квазіреальному часі. Вони роблять систему стійкою до зовнішніх впливів і дозволяють безпечно зберігати та аналізувати події незалежно від поточного стану мережі або працездатності інших компонентів.

MQTT-шина даних (транспортний рівень)

У цій архітектурі MQTT є основною транспортною шиною, що забезпечує ефективний і надійний обмін даними між усіма компонентами системи – агентами, мікросервісами, користувацькими інтерфейсами та іншими споживачами інформації. Завдяки моделі публікації-підписки MQTT дозволяє передавати події у реальному часі, без необхідності встановлювати прямі з'єднання між усіма учасниками. Це дає можливість повністю ізолювати клієнтів один від одного, спрощуючи масштабування та підвищуючи безпеку системи.

Протокол особливо добре підходить для середовищ із обмеженими ресурсами або нестабільними каналами зв'язку, оскільки має низькі накладні витрати та оптимізований для швидкої передачі невеликих повідомлень. Його використання суттєво знижує навантаження на мережу, зменшує затримки і дозволяє оперативно реагувати на події в системі.

У цій реалізації використовується захищене з'єднання через TLS, що гарантує конфіденційність та цілісність даних під час передачі. Додатково впроваджені механізми контролю доступу на рівні MQTT-топіків: кожен клієнт має права лише на ті теми, які йому дозволені, що запобігає несанкціонованому доступу до чутливої інформації.

Усі повідомлення, що генеруються агентами, – наприклад, про стан системи, активність користувача або події з моніторингу, – передаються через MQTT-брокер і далі розподіляються серед зацікавлених споживачів. Це дозволяє централізовано обробляти інформацію, вести журнал подій, реалізовувати алерти, візуалізацію або зберігання в базі даних, не навантажуючи самих агентів додатковою логікою. Такий підхід забезпечує модульність, масштабованість та простоту інтеграції з іншими системами.

Рівень обробки та аналітики

Цей рівень відповідає за глибоку обробку подій, що надходять з MQTT-брокера, та перетворення їх у зрозумілі, структуровані дані для подальшого аналізу або реакції системи. Всі повідомлення, які генеруються агентами і потрапляють на брокер, передаються в низку спеціалізованих мікросервісів, кожен з яких виконує конкретну функцію.

Сервіс обробки подій (Event Processor) є першим етапом обробки, де відбувається нормалізація структури повідомлень, об'єднання (агрегація) подій, обчислення додаткових метрик, а також фільтрація дублюючих або несуттєвих повідомлень. Завдяки цьому зменшується об'єм непотрібних даних та підвищується ефективність наступних етапів обробки.

Система правил (Rule Engine) застосовує заздалегідь визначені шаблони і логіку до потоку подій. Вона дозволяє виявляти аномалії, порушення регламентів, або інші значущі ситуації на основі заданих умов. Наприклад, перевищення порогів навантаження, підозрілу активність користувача або втрату зв'язку з критичним вузлом.

Аналітичний модуль (Analytics Engine) накопичує історичні дані та генерує звіти – щоденні, тижневі або за адаптивними сценаріями. Він підтримує інтеграцію з time-series сховищами, що дозволяє будувати тренди, візуалізувати зміни в часі, а також виявляти довгострокові патерни.

Сервіс повідомлень (Notifier) закриває цикл реагування: у випадку виявлення критичної події або порушення він автоматично розсилає сповіщення у відповідні канали. Це можуть бути повідомлення в Telegram, листи на електронну пошту, інтеграції зі Slack або іншими месенджерами. Таким чином, відповідальні особи отримують актуальну інформацію одразу після виникнення інциденту, що дозволяє зменшити час реакції і мінімізувати наслідки.

Сховище даних

Усі повідомлення, що передаються через MQTT-шину, а також їх оброблені версії, зберігаються у централізованому сховищі даних. Це сховище не є монолітним – залежно від характеру даних та аналітичних завдань використовуються різні типи баз даних, кожна з яких виконує свою функцію.

Для зберігання телеметрії, що має часову природу (наприклад, рівень навантаження системи або активність користувача з плином часу), використовується time-series база даних InfluxDB. Вона дозволяє з високою швидкістю та ефективністю будувати графіки, виявляти піки активності, а також зручно інтегрується з аналітичними панелями.

Напівструктуровані або подієві дані, що надходять від різних агентів або мікросервісів у JSON-форматі, зберігаються в NoSQL-сховищі – MongoDB. Такий підхід забезпечує гнучкість у зберіганні неоднорідних подій та можливість швидкої адаптації до нових форматів без необхідності зміни схеми бази.

Своєю чергою реляційна база даних PostgreSQL слугує основним аналітичним сховищем. Вона використовується для побудови звітів, зведеної аналітики, зберігання довідкової інформації та зв'язку між сутностями системи. PostgreSQL добре підходить для складних запитів та інтеграції з BI-системами.

Вся архітектура зберігання побудована з урахуванням масштабованості: обробка, зберігання і користувацький інтерфейс можуть розгортатися незалежно, що дає змогу ефективно використовувати ресурси та адаптувати систему до змін навантаження. Кешування подій на рівні агентів та брокера забезпечує стійкість до збоїв і тимчасових втрат зв'язку.

Основні переваги такої архітектури – це її модульність, масштабованість і гнучкість. Компоненти можна розгортати в контейнерах, що спрощує управління та оновлення. Завдяки використанню MQTT, до системи легко підключати як потужні сервери, так і малопотужні IoT-пристрої. Додаткові сервіси або інтерфейси можуть інтегруватися без серйозного втручання в наявну інфраструктуру, що робить платформу зручною для подальшого розвитку.

Висновки та перспективи подальших досліджень. У статті представлено архітектуру мікросервісної платформи моніторингу та управління інженерними системами, орієнтовану на застосування концепцій Інтернету речей (IoT). Запропоноване рішення вирізняється гнучкістю, масштабованістю та здатністю до

інтеграції із зовнішніми сервісами, що не входять безпосередньо до інфраструктури системи. Завдяки використанню MQTT як транспортного протоколу, забезпечується ефективна взаємодія між різнорідними компонентами сенсорами, виконавчими пристроями, аналітичними модулями та сервісами в хмарному середовищі.

Суттєвою перевагою платформи є її відкритість до сторонніх інформаційних джерел і сервісів, що дозволяє реалізовувати гетерогенне обчислювальне середовище для збору, зберігання, аналізу та візуалізації даних. Це формує умови для побудови адаптивних інтелектуальних систем, здатних до самонавчання, контекстного аналізу та автоматизованого прийняття рішень.

Перспективи розвитку системи пов'язані з інтеграцією методів машинного навчання та штучного інтелекту, що дозволяє автоматизовано виявляти аномалії у функціонуванні, аналізувати поведінкові патерни користувачів, прогнозувати навантаження та створювати сценарії реагування в режимі реального часу. Такі підходи також відкривають можливість для генерації інтерпретованих звітів і рекомендацій на основі глибокого аналізу отриманих даних. Обробка історичних даних з використанням time-series методів, зокрема із залученням глибоких нейронних мереж, формує підґрунтя для побудови точних предиктивних моделей. Подальші дослідження передбачають розвиток функціональності системи через впровадження потокової обробки даних, використання самонавчальних та активних навчальних алгоритмів, моделювання поведінки з опорою на багатоагентний підхід, а також розробку контекстно-орієнтованих механізмів адаптації. Усе це спрямовано на створення інтелектуального середовища керування інженерною інфраструктурою, що базується на сучасних технологіях IoT, Big Data та AI, з орієнтацією на автономність, безпеку та динамічну адаптивність системи.

References:

1. Kovács, L. and Jlidi, A. (2025), «Process Discovery for Event Logs with Multi-Occurrence Event Types», *Algorithms*, doi: 10.3390/a18020083.
2. Mileff, P. (2023), «Role of User Activity Monitoring in RPA Processes», *Proceedings of Scientific and Applied Informatics in Engineering*, doi: 10.32968/psaie.2023.1.3.
3. Linn, C., Zimmermann, P. and Werth, D. (2018), «Desktop Activity Mining - A new level of detail in mining business processes», [Online], available at: https://dl.gi.de/items/05b3d93f-e19d-40f7-980b-67bace8a21e7
4. Martínez-Rojas, A., Alonso-Rocha, J.L., Jimenez Ramírez, A. and González Enríquez, J. (2024), «From Screenshots to Process Models: Improving Activity Identification Through Screen Text», doi: 10.1007/978-3-031-70445-1_8.
5. Agostinelli, S. and Marrella, A. (2022), «Intelligent Robotic Process Automation: Generating Executable RPA Scripts from Unsegmented UI Logs», [Online], available at: https://ceur-ws.org/Vol-3310/paper13.pdf
6. Rybinski, F. and Schüler, S. (2022), «Process Discovery Analysis for Generating RPA Flowcharts», doi:10.1007/978-3-031-16168-1_15.
7. Wil, M.P. van der Aalst (2022), «Process Mining and RPA: How To Pick Your Automation Battles?», [Online], available at: https://www.researchgate.net/profile/Wil-Aalst/publication/357933625_Process_Mining_and_RPA_How_To_Pick_Your_Automation_Battles/links/61e82caa9a753545e2e0f0d0/Process-Mining-and-RPA-How-To-Pick-Your-Automation-Battles.pdf
8. Faiz, M.M.U. and Patil, N.(2023), «Automated Robotic Processes that Learn on Their Own», *Int. Conf. on PEEIC*, doi: 10.1109/PEEIC59336.2023.10452060.
9. Park, A., Jung, S., Yune, I. and Lee, H.Y. (2025), «Applying Robotic Process Automation to Monitor Business Processes in Hospital Information Systems», *JMIR Med Inform.*, doi: 10.2196/59801.
10. Marciniak, P. (2024), «Business robot farm management in the Robotic Process Automation (RPA)», *Management*, doi: 10.58691/man/190214.

Воротніков Володимир Володимирович – доктор технічних наук, доцент, професор кафедри комп'ютерної інженерії та кібербезпеки Державного університету «Житомирська політехніка».

https://orcid.org/0000-0001-8584-3901.

Наукові інтереси:

- комп'ютерні мережі та мережні технології;
- мережна безпека, кібербезпека;
- керування складними інформаційними системами.

E-mail: kkik_vvv@ztu.edu.ua.

Шелуха Олексій Олегович – кандидат технічних наук, доцент кафедри комп'ютерної інженерії та кібербезпеки Державного університету «Житомирська політехніка».

https://orcid.org/0000-0002-6088-8262.

Наукові інтереси:

- комп'ютерні системи та мережі;
- інтернет речей;
- системи автоматизованого управління та інформаційно-вимірювальні системи.

E-mail: kkik_shoo@ztu.edu.ua.

Матвєєв Костянтин Ігорович – старший викладач кафедри комп'ютерної інженерії та кібербезпеки Державного університету «Житомирська політехніка».

https://orcid.org/0009-0004-4594-6956.

Наукові інтереси:

- індустріальний інтернет речей (IIoT);
- автоматизація технологічних процесів;
- аналіз даних та системи моніторингу;
- комп'ютерно-інтегровані технології, розумні будівлі, системна інженерія.

E-mail: kakit_mki@ztu.edu.ua.

Vorotnikov V.V., Shelukha O.O., Matvieiev K.I.

User activity monitoring and automated office IT infrastructure management system

This article presents the concept, architecture, and implementation of an intelligent microservice platform for monitoring user activity and automating the management of office IT infrastructure. In the context of rapid digital transformation and growing cybersecurity threats, the platform addresses the need for transparency, anomaly detection, and adaptive response across complex IT environments. A novel aspect of the proposed approach is the conceptualization of each personal computer as an active Internet of Things (IoT) node within a unified digital ecosystem.

The system is based on a multi-level, event-driven IoT architecture that employs an MQTT bus as the transport layer to ensure scalability, flexibility, and fault tolerance. It consists of three core layers: data collection agents, a message broker, and analytical-control microservices. These components enable behavior-based analytics, detection of unauthorized software, identification of third-party or potentially malicious processes (e.g., cryptominers), and real-time risk prediction. The platform integrates secure communication protocols, encryption, event caching, and mechanisms for privacy protection and data persistence during connectivity loss.

A detailed implementation of a Python-based software agent is provided, capable of gathering technical and behavioral parameters such as CPU load, user interaction, and network status. The architecture supports seamless integration with cloud-based analytics tools, AI modules, and cybersecurity systems, facilitating dynamic policy adjustment and proactive incident response.

The platform is designed to incorporate self-learning mechanisms, contextual awareness, and multi-agent coordination–supporting the transition from conventional monitoring tools to an adaptive, intelligent infrastructure aligned with Industry 4.0 paradigms, where each workstation functions as a smart sensor-actuator unit in the organizational IT landscape.

Keywords: intelligent system; user behavior monitoring; automated IT management; microservice architecture; MQTT; agent-based analytics; cybersecurity; IoT workstations; anomaly detection; Industry 4.0; real-time infrastructure adaptation.

Стаття надійшла до редакції 18.04.2025.