Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Кузьмін К.К.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Кузьмін К.К. Дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віддаленого контролю на основі технології Intel ME.

Метою розробки є дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME.

Об'єктом дослідження є процес віддаленого контролю на основі технології Intel ME.

Предметом дослідження є методи віддаленого контролю на основі технології Intel ME.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи віддаленого контролю на основі технології Intel ME.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, віддаленого контролю, Intel ME

ABSTRACT

Kuzmin K.K. Research and software implementation of a remote control system based on Intel ME technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for a remote control system based on Intel ME technology.

The purpose of the development is the research and software implementation of a remote control system based on Intel ME technology.

The object of the research is the process of remote control based on Intel ME technology.

The subject of the research is remote control methods based on Intel ME technology.

The research methods are based on the methods of the theory of computer network construction, methods of mathematical statistics, and methods of software development.

The result of the work is the software implementation of a remote control system based on Intel ME technology.

In the process of working on the software model, an analysis of existing hardware and software tools was performed. All components of the developed software are fully described.

A user-friendly user interface has been developed. Instructions for working with the software are provided.

The program can be used on a PC with Windows 10/11.

The program is developed in the Python environment.

Keywords: computer engineering, remote control, Intel ME

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЛОМ	–	локальна обчислювальна мережа
ПК	–	персональний комп’ютер
ПЗ	–	програмне забезпечення
DNS	–	Domain Name System
http	–	HyperText Transfer Protocol
IP	–	Internet Protocol

ВСТУП

Актуальність теми. Технологія Intel ME (або AMT, Active Management Technology) є одним із самих загадкових і потужних елементів сучасних x 86-платформ. Інструмент споконвічно створювалося як рішення для віддаленого адміністрування. Однак він має настільки потужну функціональність і настільки непідконтрольний користувачам Intel-based пристроїв, що багато хто з них хотіли б відключити цю технологію, що зробити не так-те просто.

Підсистема Intel Management Engine (ME) являє собою додатковий «схований» процесор, що є присутнім у всіх пристроях на базі чипсетів Intel (не тільки в PC і ноутбуках, але й у серверах).

Середовище виконання ME ніколи не «спить» і працює навіть при виключеному комп'ютері (при наявності чергової напруги), а також має доступ до оперативної пам'яті, мережного інтерфейсу, USB контролера й убудованого графічного адаптера.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем віддаленого контролю на основі технології Intel ME.

– Дослідження системи віддаленого контролю на основі технології Intel ME.

– Програмна реалізація системи віддаленого контролю на основі технології Intel ME.

Об'єктом дослідження є процес віддаленого контролю на основі технології Intel ME.

Предметом дослідження є методи віддаленого контролю на основі технології Intel ME.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод віддаленого контролю на основі технології Intel ME.

– Розроблено вітчизняний продукт віддаленого контролю на основі технології Intel ME, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віддаленого контролю на основі технології Intel ME.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2025 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Intel Management Engine (ME) – підсистема, що убудована в усі сучасні комп'ютерні платформи (десктопи, лептопи, сервери, планшети) із чипсетами компанії Intel. Ця технологія багатьма сприймається як апаратна «закладка», і на тобто причини. Досить сказати, що Intel ME є єдиним середовищем виконання, що:

– працює навіть тоді, коли комп'ютер виключений (але електроживлення подається);

– має доступ до всього вмісту оперативної пам'яті комп'ютера;

– має позаполосний доступ до мережного інтерфейсу.

Ошелешений присутністю такого компонента в комп'ютері, користувач (виходить, що саме «користувач», а не «власник») напевно задавався питанням: чи можна виключити Intel ME?

Нагадаємо, що основним компонентом Intel ME є убудований у чипсет мікроконтролер з кастомною архітектурою. Відома лише базова модель, це 32-х розрядний ARCtangent-A4 (ME 1.x – 5.x), ARCtangent-A5 (ME 6.x – 10.x), SPARC (TXE) або x86 (ME 11.x – ...).

Починаючи з 6-й версії, ME-контролер вбудовують в усі чипсети Intel.

Завантажник його прошивання зберігається у внутрішньої ROM і недоступний для аналізу. Саме прошивання розташовується в регіоні ME у зовнішньої SPI флеш-пам'яті (тобто у тій же пам'яті, де зберігається BIOS). Структура цього прошивання така, що весь здійсненний код розбитий на модулі, які зберігаються в стислому виді (або кастомною реалізацією алгоритму Хаффмана, або LZMA). Ці кодові модулі криптографічно захищені від модифікацій.

Отже, розглянута підсистема є апаратно-програмною основою для різних системних функцій (деякі раніше реалізовували в BIOS) і технологій Intel. Їхня імплементація включається до складу прошивання Intel ME. Однією з таких технологій, що використовують кілька особливих привілеїв Intel ME, є Active Management Technology (AMT).

1.2 Область застосування

Областю застосування є віддалений доступ до комп'ютера. Віддалений доступ – функція, що дає користувачеві можливість підключатися до комп'ютера за допомогою іншого пристрою через інтернет практично звідсіля. Користувач працює з файлами й програмами точно так само, як якби він перебував біля цього комп'ютера. Особливо придасться ця функція тим компаніям, де більшість співробітників перебуває за межами офісу, на частковому фрилансі, аутсорсингу або у відрядженнях, але при цьому вони мають потребу у відновленні робочої інформації, перегляді корпоративної пошти та ін. Їм не потрібно буде завантажувати всі необхідні для роботи дані на зовнішній носій або відправляти їх поштою – досить зв'язатися з офісним комп'ютером.

Віддалений доступ використовують системні адміністратори для керування системою й усунення збоїв у її роботі, і керівники, що бажають проконтролювати процес виконання завдання своїми підлеглими. Застосовується він і для дистанційного навчання в освітніх установах.

На зорі розвитку технології користувач на своєму ноутбуці фактично міг тільки бачити екран віддаленого комп'ютера. Зберегти дані, наприклад, можливості не було. Пропускна здатність телефонних ліній, що використовувалися для віддаленого доступу, була вкрай низкою.

Варіанти організації віддаленого доступу

Відомо два варіанти організації віддаленого доступу:

– установка спеціалізованого ПЗ на власному сервері;

– договір із хмарним провайдером по моделі DaaS (desktop as a service) – віртуальний робітник стіл.

Розглянемо кожний з варіантів докладніше, щоб зрозуміти їхні переваги й недоліки.

Програми для віддаленого доступу: «серверний варіант»

Для створення віддаленого підключення використовують спеціальні програми. Обов'язкова умова – наявність постійного доступу в інтернет, комп'ютерів, що володіють певними характеристиками й сервера. Віддалене підключення зв'язує два робітників станції через інтернет. У стандартному додатку Windows з'єднання відбувається між двома IP-адресами, але якщо комп'ютер перебуває в локальній мережі, то підключитися до нього ззовні можна тільки за допомогою спеціальних програм віддаленого доступу. Таке ПЗ уможливлює підключення до іншого комп'ютера з будь-якої точки світу. Програми дозволяють бачити робочий стіл і виконувати всі дії на віддаленому пристрої, змінювати налаштування ПЗ, обмінюватися файлами, робити принт-скріни, шифрувати передані дані, проводити конференції, підключати веб-камери, віддалені проектори та інші мережні пристрої.

Мінусом цього типу підключення є вимога до наявності спеціальних знань по налаштуванню офісних маршрутизаторів. Крім того, при такому способі організації доступу підвищується ймовірність злому й проникнення в локальну мережу сторонніх осіб.

DaaS / VDI – рішення під ключ: «хмарний варіант»

Послуга дає клієнтам доступ до готового до роботи віддаленого робочого столу. У нього входить набір певних програм, що розширюється залежно від потреб користувача. Працювати можна з офісних або домашніх комп'ютерів, ноутбуків і мобільних пристроїв: робітник стіл скрізь буде однаковий. Операційні дії виконуються на віддаленому сервері постачальника послуг, тому вимоги до пристроїв клієнта мінімальні (тонкі клієнти). При цьому варіанті дані й програми користувача розміщаються не на локальному сервері, а в хмарі. Доступ до них

можливий з будь-якого місця, де є інтернет. Для початку роботи буде потрібно укласти договір з постачальником послуги, попередня установка якого-небудь ПЗ не потрібна. Провайдери гарантують безпечне зберігання інформації в дата-центрах і захист від хакерских атак і вторгнення вірусів.

Недолік послуги тільки один – залежність від якості каналу зв'язку.

Підключення до віддалених робочих столів у хмарі в перспективі може стати повсюдним. Компанії будуть урятовані від необхідності закуповувати дороге встаткування, витрачати гроші на його обслуговування й відновлення ПЗ. Максимум ресурсів робочому столу можна забезпечити буквально двома кліками миші. При цьому якість роботи в хмарі й безпека даних постійно зростає. Наприклад, важлива бухгалтерська інформація компанії зберігається не на місцевому сервері, а в захищеному сховищі й регулярно, на додаток до цього резервується в іншому, географічно віддаленому даті-центрі. Навіть у випадку віддалення сервера податковими або правоохоронними органами всі дані, необхідні для якнайшвидшого поновлення роботи, дуже легко й швидко відновити. При втраті ноутбука зашифрована бізнес-інформація, що перебуває в ньому, також буде централізовано захищена, а значить – гарантовано не потрапить у руки, наприклад, конкурентів. І навіть якщо щось раптом відбудеться з робочим місцем співробітника, наприклад, він проллє на клавіатуру чай або «спалить» материнську плату від перепаду напруги в електромережі, процес відновлення пройде дуже швидко – всі додатки, їхні налаштування й інформація також зберігаються на віддаленому сервері.

Що важливо врахувати при виборі провайдеру DaaS?

На ринку з'являється усе більше постачальників подібних послуг, рядовому користувачеві зробити вибір з кожним днем сутужніше. Я б порекомендував звернути увагу на дві речі: забезпечення рівня безпеки інформації й наявність власних розробок. Перше говорить про відношення провайдеру до своїх клієнтів, а друге – про рівень кваліфікації співробітників.

Приватні дата-центри приділяють питанням захисту інформації першорядна увага. Всі дані клієнта зберігаються в безпечній хмарі, створеній на базі власних дата-центрів у декількох містах світу. Строго дотримується безпека при роботі з даними з особистих пристроїв користувачів.

Віддалений доступ до даних, розташованим у хмарі, забезпечується у відповідності зі стандартами зберігання й обробки даних кредитних карт PCI DSS. Клієнтам пропонується 4 штатні конфігурації із передвстановленими програмами. Можлива й налаштування персональних конфігурацій по вимогах користувача.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Контроль за станом AMT

AMT – технологія віддаленого адміністрування комп'ютерних систем, для яких заявлена офіційна підтримка Intel vPro (це бренд, що поєднує кілька технологій, у тому числі, AMT). Мова йде про системи із чипсетами лінійки Q (наприклад, Q57 або Q170).

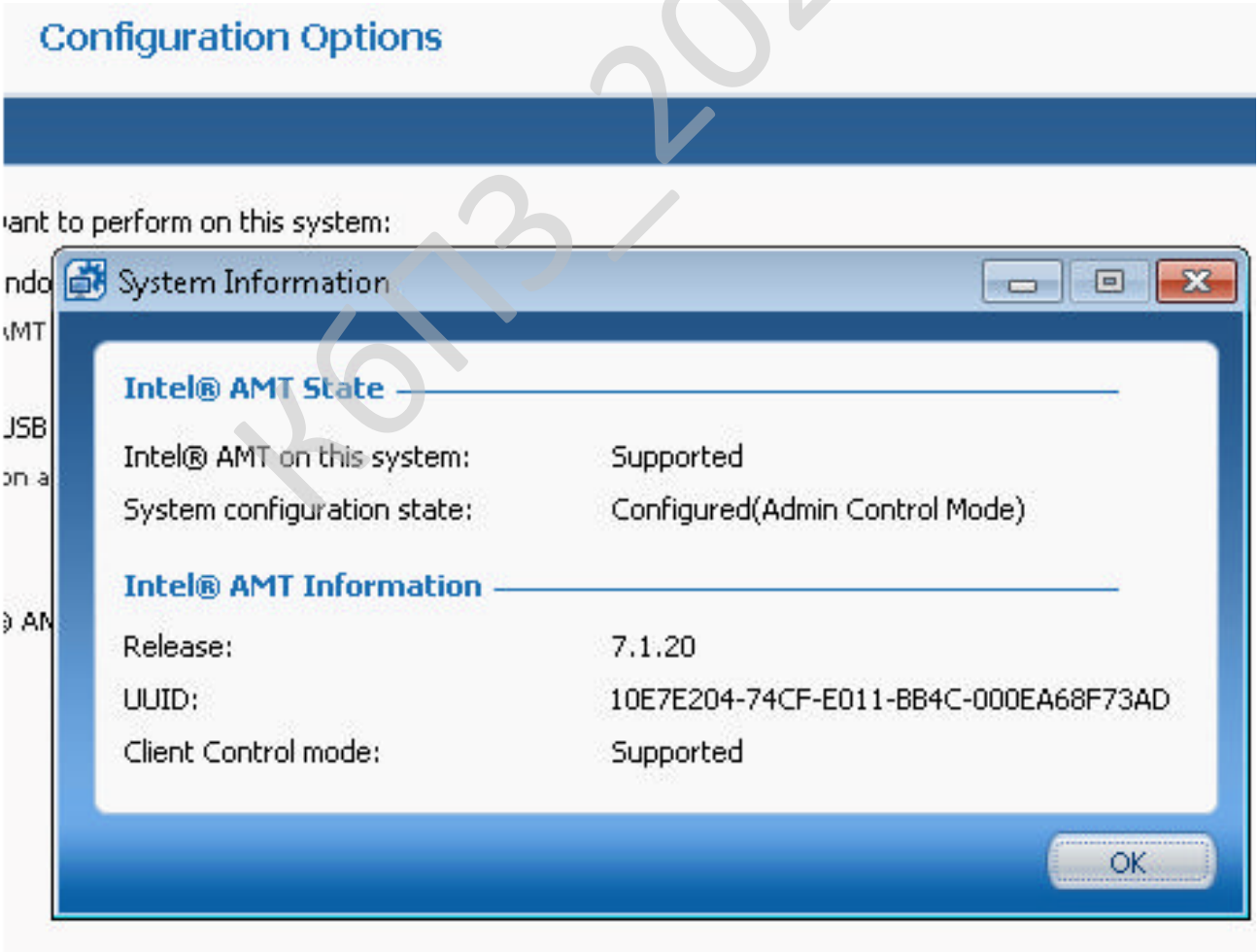

Рисунок 2.1 – Інтерфейс користувача ACUwizard

З огляду на високу вартість таких платформ, навряд чи хтось випадково придбає комп'ютер з AMT для того, щоб принципово цією технологією не користуватися. Проте, якщо під рукою саме такий продукт, і їсти необхідність переконатися, що AMT на сучасний момент виключений, варто скористатися утилітою ACUwizard.

Або засобом Intel Management and Security Status (входить до складу ПЗ Intel ME для vPro-платформ, можна виявити в треї).

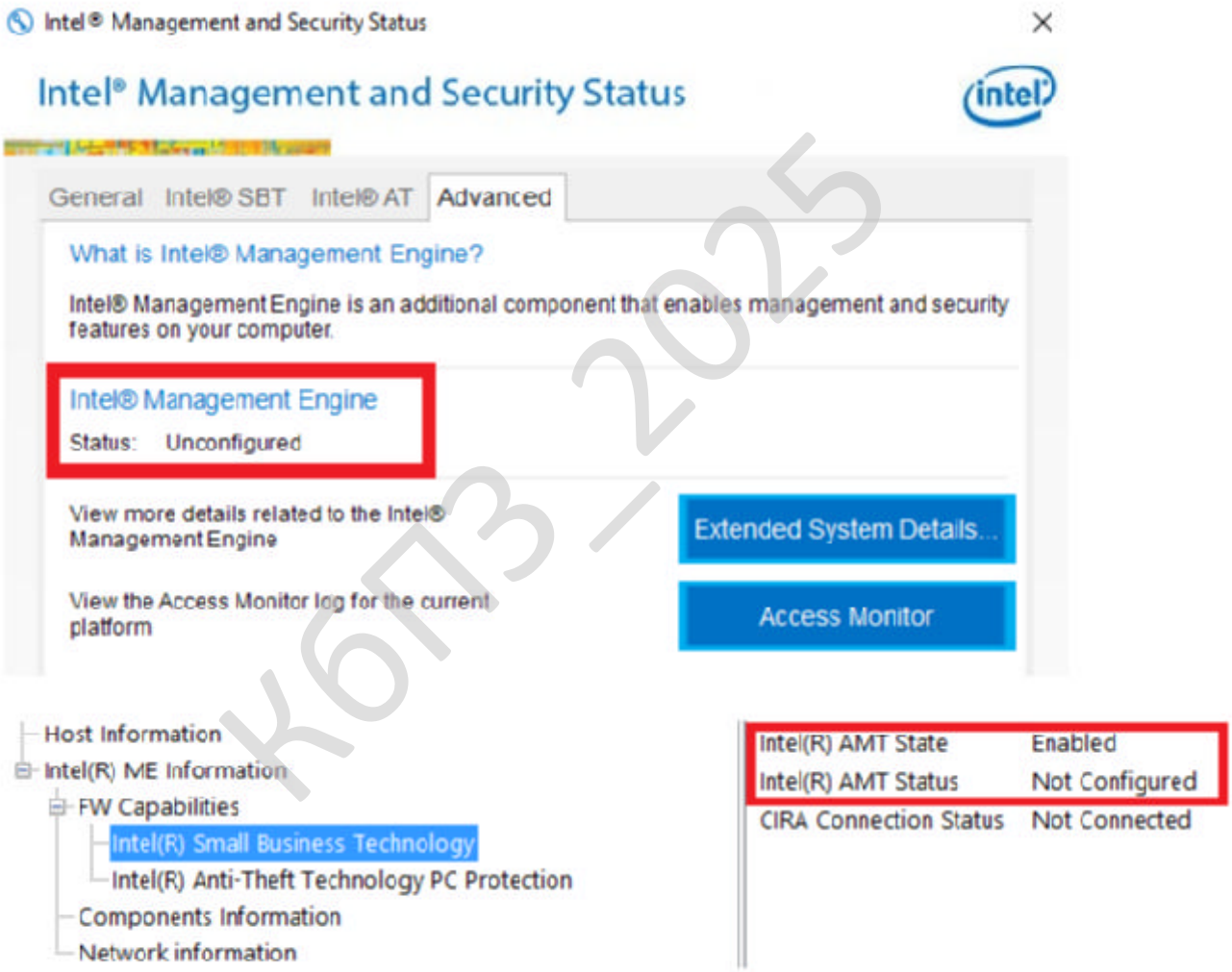

Рисунок 2.2 – Інтерфейс користувача Intel Management and Security Status

Нарешті, щоб захистити комп'ютери у своїй мережі від несанкціонованого керування ззовні, необхідно настроїти зовнішній файрвол на фільтрацію AMT-Запитів за ознаками. Явною ознакою AMT-Запиту може бути порт, до якого відбувається обіг:

– 5900 – AMT VNC-сервер без шифрування;

– 16992 – AMT веб-сервер за протоколомі HTTP;

– 16993 – AMT веб-сервер за протоколомі HTTPS;

– 16994 – AMT redirection для SOL, IDE-R, KVM без шифрування;

– 16995 – AMT redirection для SOL, IDE-R, KVM з TLS.

У продуктах, що не ставляться до розряду vPro-платформ AMT включити не можна, однак до складу прошивання Intel ME входять мережні драйвери.

Це означає, що ME-контролер (не будемо забувати, що він завжди включений) має технічну можливість використання мережного інтерфейсу.

Тому проблему варто вирішувати ґрунтовно – намагатися виключити підсистему Intel ME.

Вимикання Intel ME за допомогою утиліт з Intel System Tool Kit

Вендорам материнських плат компанія Intel надає:

– Прошивання Intel ME у бінарному виді.

– Модулі MEBx для BIOS.

– ПЗ Intel ME для ОС.

– Intel System Tool Kit (STK) – комплект програмних засобів і документації для складання образів SPI флеш-пам'яті, застосування цих образів і одержанні інформації про поточний стан Intel ME.

Незважаючи на те, що цей комплект поширюється по NDA (судячи з міток «Intel Confidential» у прикладених документах), багато хто вендори забувають його вирізати з архіву з ПЗ Intel ME, що передається користувачам. А ще не закривають свої ftp-сервери від зовнішнього доступу. У результаті, що витекли версій STK дуже багато. Тут можна злити комплект для будь-якої версії Intel ME.

Важливо, щоб major і minor version (перша й друга цифри) використовуваного STK збігалася з версією Intel ME цільової системи, інформацію про яку можна одержати, наприклад, скориставшись ME analyzer.

Перевіряти поточний стан Intel ME можна за допомогою утиліти MEinfo, що через Management Engine Interface (MEI) одержує інформацію про роботу цієї підсистеми.

Нагадаємо, що MEI є інтерфейсом для зв'язку основного CPU з підсистемою Intel ME і являє собою набір регістрів у конфігураційному просторі PCI і в MMIO. Команди цього інтерфейсу не документовані, як і сам протокол.

Flash Image Tool

На старих платформах (Intel ME версії 5.x і нижче) виключити дану підсистему можна, скориставшись Flash Image Tool (утиліта з STK, призначена для складання образів SPI флеш-пам'яті з окремо взятих регіонів BIOS, ME, Gb). При складанні задаються параметри, які прописуються в цих регіонах і в регіоні Flash Descriptors. В останньому є один дуже цікавий прапор, «ME disable»:

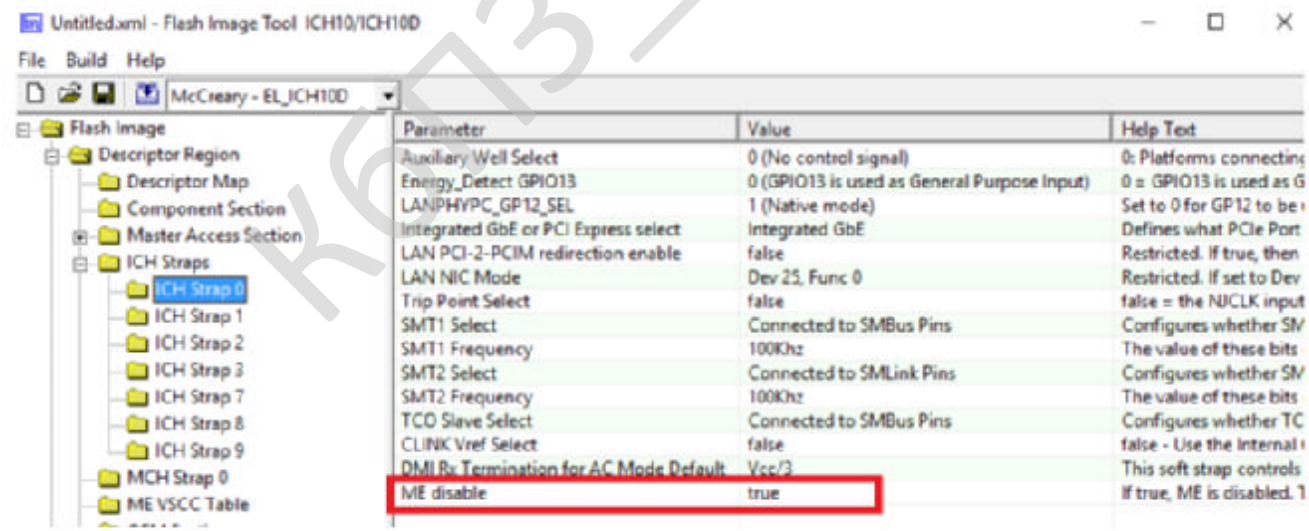

Рисунок 2.3 – Інтерфейс користувача Flash Image Tool

Таким чином, для вимикання Intel ME на цільовій комп'ютерній системі, у її SPI флеш-пам'ять варто записати (програматором) новий образ із виставленим прапором «ME disable».

Чи працює цей спосіб, нам невідомо. Але звучить правдоподібно, з огляду на, що ME-контролер у тих версіях інтегрувався тільки в чипсети лінійки Q, тобто був не обов'язковим компонентом для всіх платформ.

Flash Programming Tool

Починаючи з Intel ME 9 версії, в утиліту Flash Programming Tool, призначену для програмування SPI флеш-пам'яті комп'ютерних платформ, була додана можливість тимчасово виключати Intel ME.

Вимикання виконується відправленням команди в MEI. Після відпрацьовування Intel ME не подає «ознак життя», не відповідає навіть MEI:

Відповідно до документації, у такому стані підсистема Intel ME перебуває до наступного запуску комп'ютера або перезавантаження.

На vPro-платформах можливість відправлення цієї команди є й у більше ранніх версіях. Для цього необхідно скористатися розділом конфігурування ME / AMT в BIOS, де повинна бути опція «Intel ME disable»:

Не можна говорити про те, що цей спосіб дозволяє повністю відключити Intel ME, хоча б тому, що до моменту прийняття команди на відключення ME-контролер встигне завантажитися, а виходить, виконати деяку частину коду прошивання.

Незважаючи на те, що Intel ME не подає «ознак життя» після цієї операції, невідомо, чи може цю підсистему заново включити який-небудь сигнал ззовні. Також неясно, наскільки Intel ME виключена.

Примусове вимикання Intel ME

В інтересах виключення можливості виконання ME-контролером коду прошивання, логічно спробувати обмежити йому доступ до неї. А що? Ні коду – немає проблеми.

Рисунок 2.4 – Інтерфейс користувача Flash Programming Tool

Проаналізувавши документацію, що додається до STK, і, небагато подумавши, ми припустили, що це можна зробити такими способами.

1. Вирізати (обнулити) ME регіон з SPI флеш-памяти. Ті, хто пробував так робити повідомляють про те, що їхня платформа або не завантажувалася без наявності справжнього прошивання ME, або вимикалася рівно після 30 мінут роботи.

Відмова комп'ютерної системи вантажитися без прошивання Intel ME можна пояснити важливістю ME-контролера в процесі ініціалізації апаратної складової. А 30-хвилинний таймаут наводить на думку про WDT (Watch Dog Timer).

2. Включити не дескрипторний режим SPI флеш-пам'яті, тобто «по старинці» коли в ній утримувався тільки BIOS. Для цього потрібно зробити одне із двох:

– стерти перші 0x20 байт у ній (у такий спосіб ушкодивши сигнатуру 0x0FF0A55A, що визначає режим роботи флеш-пам'яті);

– виставити джампер HDA_SDO (якщо він є).

Таким чином, ME-контролер не одержить доступ до свого регіону, і, отже, не буде виконувати прошивання.

З одного боку, ME-контролер так само, як і в попередньому випадку, може перешкоджати нормальній роботі комп'ютерної системи. З іншого боку, не дескрипторний режим включає т.зв. manufacturing mode, що використовується вендорами в відладочних цілях, і є шанс, що система запуститься.

3. Відомо, що прошивання Intel ME розпаковується у виділену й сховану від основного CPU область оперативної пам'яті – ME UMA. Виділення й блокування цієї області здійснює BIOS під час конфігурування карти пам'яті. Тоді чому б не вирізати ці фрагменти коду з BIOS, щоб дана область не виділялася. Тоді прошивання ME не буде розпаковуватися й виконуватися.

Проведені експерименти показали, що такий спосіб теж не годиться, і система не запускається. До того ж, в ME-контролера є внутрішня SRAM, що використовується при неприступності ME UMA. Тому частина прошивання однаково буде виконуватися.

Ми розповіли про передбачуваний і потребуючий розвиток способах вимикання Intel ME:

– відключення при кожному старті командою в MEI або відключення прапором у регіоні flash descriptors (залежно від версії);

– обмеження доступу ME-контролеру до свого прошивання або переклад комп'ютерний платформи в manufacturing mode.

– перешкоджання нормальній роботі ME-контролера не забезпечуючи його runtime-пам'яттю.

Очевидно, що деякі запропоновані рішення спричиняють непрацездатність комп'ютерної системи, інші не дають ніякої гарантії того, що підсистема Intel ME дійсно виключена. У зв'язку із цим, ми прийшли до виводу, що повністю виключити Intel ME украй складно.

Імовірно, це пов'язане з тим, що, відключаючи Intel ME, ми нейтралізуємо важливий компонент в архітектурі комп'ютерної системи. Наприклад, без ME комусь буде вирішувати важливі системні завдання начебто ACPI або ICC (які

колись реалізовувалися в BIOS). Щоб змусити платформу стабільно працювати без ME, як мінімум, необхідно повернути реалізацію цих технологій в BIOS.

Так чи інакше, питання про те, як відключити Intel ME без втрати працездатності комп'ютерної системи, залишається відкритим.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи віддаленого контролю на основі технології Intel ME.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Незважаючи на настільки великі можливості Intel Management Engine (ME), існують питання до рівня захищеності ME – раніше дослідники вже знаходили серйозні уразливості й вектори атак. Крім того, підсистема містить потенційно небезпечні функції – віддалене керування, NFC, схований сервісний розділ (hidden service partition). Інтерфейси підсистеми ME недокументовані, а реалізація закрита.

Всі ці причини приводять до того, що багато хто розглядають технологію ME у якості «апаратної закладки». Ситуацію збільшує той факт, що з однієї сторони в користувача пристрою немає можливостей по відключенню цієї функціональності, а з інший виробник устаткування може допустити помилки в конфігурації ME.

Гарна новина полягає в тому, що способи відключення ME все-таки існують.

Техніки відключення Intel ME

Фахівці описали трохи технік відключення даної підсистеми:

– Засновані на збої ініціалізації ME.

– Через механізм відновлення мікропрограми ME.

– Недокументовані команди.

– Недокументований механізм, призначений для розроблювачів апаратури – Manufacture Mode.

Дослідники встановили, що розроблювачі апаратних платформ часто забувають виключати режим Manufacture Mode, що дозволяє використовувати останній метод на великій кількості комп'ютерів без яких або додаткових витрат у режимі «реального часу».

Більшість методів відключення використовують убудовані механізми ME, розроблені для вендорів пристроїв на платформі Intel.

І проте, виникає резонне питання: « чиДійсно ME перестає працювати в повному обсязі при використанні її убудованих механізмів відключення?» Як доказ факту відключення ME дослідники приводять наступний аргумент: ME працює у двох режимах використання пам'яті: тільки SRAM (убудований в ME) і SRAM + UMA. UMA – це частина пам'яті хоста, що використовується пам'ять, що підкачується як (swap). Після ініціалізації DRAM-контролера хостом ME завжди перемикається в режим SRAM + UMA.

Таким чином, якщо ME дійсно виключена, то при відключенні на апаратному рівні доступу ME до UMA-Пам'яті в довільний момент (засобами каналу VCm), у ME не буде відбуватися апаратних збоїв, пов'язаних з відсутністю даних і коду, які були витиснуті в UMA пам'ять (такі апаратні збої приводять до аварійного відключення живлення з основних апаратних компонентів платформи). З іншої сторони застосування цих методів дозволяє здійснити DoS-атаки на технологію AMT у випадку її застосування для віддаленого керування.

3.2 Розробка структурної схеми

В 2005 році компанія Intel представила Active Management Technology (AMT) версії 1.0 – рішення для віддаленого адміністрування (керування, інвентаризація, відновлення, діагностика, усунення неполадок і т.д.) і захисту десткопних комп'ютерних систем, свого роду аналог технології Intelligent Platform Management Interface (IPMI), що використовується в серверах.

Архітектура AMT 1.0 ґрунтується на інтегрованому в чипсет мікроконтролері (Management Engine), наділеному досить вражаючими можливостями, наприклад:

– позаполосний (out-of-band) доступ до мережного інтерфейсу (Ethernet), що він розділяє з основним CPU, але, маючи власний контролер канального

рівня, здійснює моніторинг усього вхідного мережного трафіку, з якого «вирізує» (за допомогою Packet Filter) пакети, призначені для нього. Для ОС (наявність і стан якої, до речі, на роботу AMT ніяк не впливає) цей трафік уже не видний;

– внутрішній веб-сервер з TLS-шифруванням;

– доступ до периферійного встаткування, одержання й зберігання в енергонезалежній пам'яті (там же, де і його прошивання) інформації про нього.

А ще цей мікроконтролер починає працювати при подачі живлення на материнську плату комп'ютерної системи (тобто при підключенні комп'ютера до електричної мережі, ще до того, як користувач натисне кнопку Power).

Отже, Management Engine завжди включений, але використання можливостей AMT вимагає активації (має на увазі завдання пароля, мережних параметрів,...) в BIOS setup, а точніше в MEBx setup.

Похвально, що дефолтний пароль («admin») при першому вході обов'язково потрібно змінити на новим, задовольняючим певним вимогам: мінімум 8 символів, серед яких повинні бути присутнім хоча б одна цифра, одна заголовна буква й один спец. символ.

Після налаштування AMT-сумісної комп'ютерної системи, віддаленому адміністраторові стають доступними мережні функції (для їхнього використання потрібен уведення логіна й пароля):

– інвентаризація апаратного забезпечення;

– веб-інтерфейс (по HTTP через порт 16992);

– Serial Over LAN (SOL) – віртуальний COM-Порт через мережу, що дозволяє включати / перезавантажувати / виключати комп'ютер, одержувати доступ до меню BIOS setup;

– IDE-Redirection (IDE-R) – опція перенапрямку завантаження з локального завантажувального пристрою на віддалене (попередньо підготовлений образ системи).

AMT 1.0 була реалізована на інтегрованому в південний міст чипсета (Input / Output Controller Hub, ICH) мережному модулі Intel 82573E series Gigabit Ethernet Controller.

Потім, в 2006 році, починаючи з AMT версії 2.0, мікроконтролер перенесли в північний міст чипсета (Graphics and Memory Controller Hub, GMCH). Саме тоді підсистему найменували в Intel Management Engine (ME) версії 2.0.

Одночасно із цим з'явився бренд Intel vPro, що позначав комплекс реалізованих на основі Intel ME технологій: AMT, Trusted Execution Technology (TXT) і Virtualization Technology (VT). Пізніше в цей список увійшли Identity Protection Technology (IPT) і Anti-Theft (AT).

Тоді ж Intel ME наділили ще більшою кількістю вражаючих можливостей, серед яких – повний доступ до всього вмісту оперативної пам'яті комп'ютера через внутрішній DMA-контролер, а надалі з'явилася можливість моніторингу відеопотоку, що виводиться на монітор (правда, тільки у випадку використання убудованого графічного ядра).

Поступово на цю підсистему стали навішувати усе більше різних системних функцій (деякими раніше займався BIOS) для забезпечення працездатності комп'ютерної платформи:

– частина функцій Advanced Control and Power Interface (ACPI) і Alert Standard Format (ASF);

– Quiet System Technology (QST);

– Integrated Clock Control (ICC);

– Trusted Platform Module (TPM);

– і інших технологій.

AMT теж не стояла на місці й активно розвивалася: змінювався состав використовуваних протоколів (наприклад, додалася підтримка HTTPS через порт 16993), у версії 6.0 для віддаленого адміністратора з'явилася фіча Remote Desktop, вона ж KVM (Keyboard Video Mouse), та інше.

Проте, через високу вартість реалізації, ця підсистема була присутня, за декількома виключеннями, тільки на материнських платах із чипсетами Intel лінійки Q.

Тоді до чого вся ця специфіка заліза із шильдиком vPro, що мало хто здобував через високу вартість (ну й інших причин)?

Справа в тому, що, починаючи з 2010 року, разом з переносом частини функціональних блоків північного мосту (графічне ядро, контролер пам'яті, ...) у корпус CPU, підсистему Intel ME стали вбудовувати в усі чипсети виробництва Intel. При цьому ME-контролер залишився в корпусі чипсета – в Platform Controller Hub (PCH). Це чипсети 5 серії й вище.

Функціональність AMT донині залишається доступною тільки на чипсетах лінійки Q, тобто тільки на встаткуванні із шильдиком vPro.

Думаєте тільки десктопи й ноутбуки? Ні, Intel-а відповідь!

Та ж доля осягла й серверні платформи від Intel: підсистема убудована в них, але під іншим ім'ям – Intel Server Platform Services (SPS). Відбулася поява й в SoC (on-a-Chip) під ім'ям Intel Trusted Execution Engine (TXE).

У підсумку архітектура кожної сучасної мобільної / лаптопної / дескопної / серверної комп'ютерної платформи із чипсетом / SoC від Intel містить у собі саму потайливу (від користувача системи) і привілейоване середовище виконання – підсистему Intel ME. Не дивно, що розробляючи цю архітектуру, компанія Intel була змушена серйозно попрацювати над її захистом від компрометації.

Розглянемо архітектуру цієї підсистеми, щоб розібратися в застосованій моделі безпеки.

Архітектура Intel ME

Intel Management Engine (ME) – убудована в комп'ютерні платформи підсистема, що забезпечує апаратно-програмну підтримку різних технологій Intel.

Як уже було сказано, перші версії цієї підсистеми були засновані на двокорпусних чипсетах Intel. Тоді як базова модель ME-контролера використовувався ARCtangent-A4 зі стандартною системою команд ARC32.

В однокорпусних чипсетах уже використовувалися ARCtangent-A5 / ARC600 з компактною системою команд ARCompact (ARC16/32).

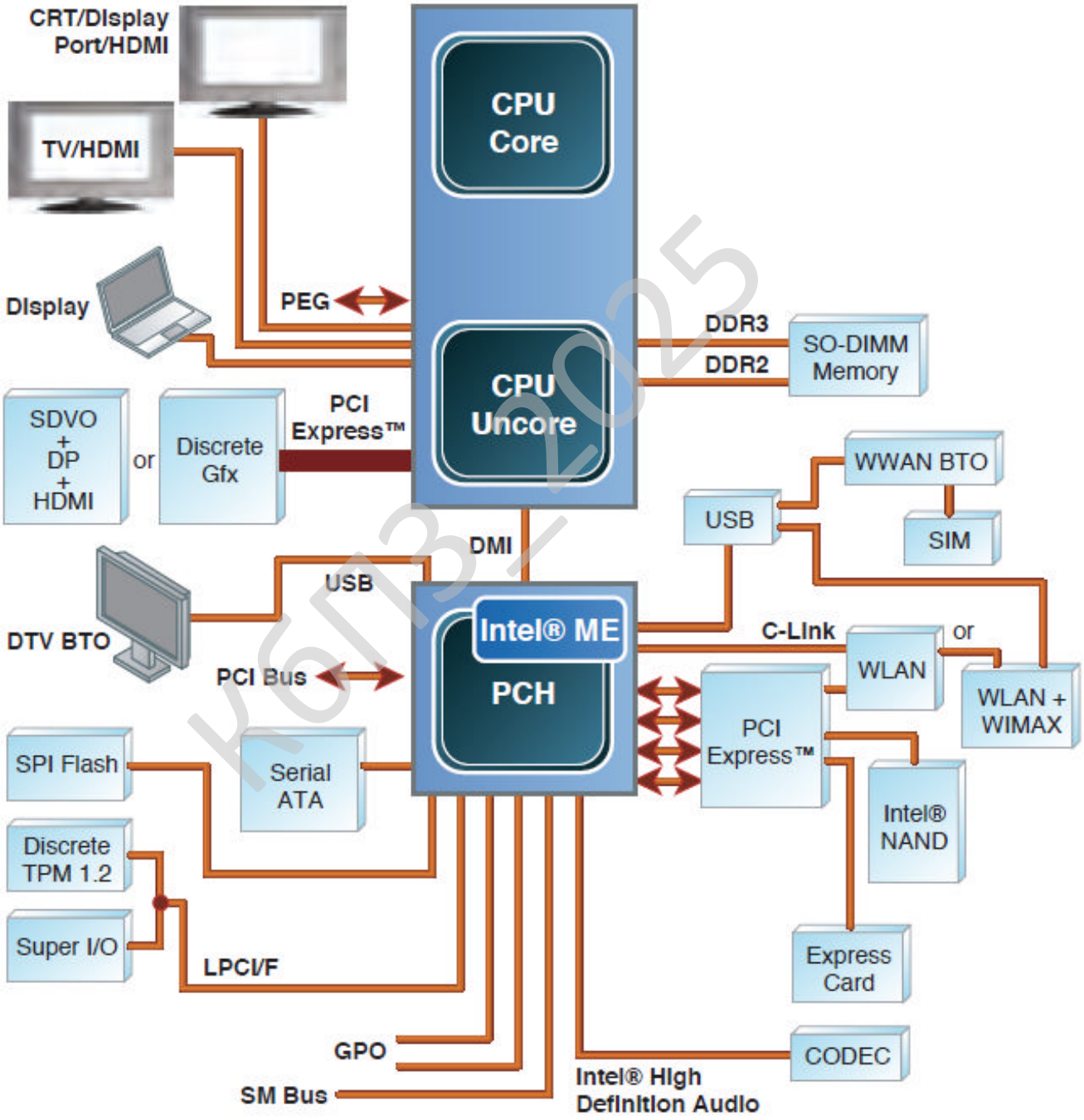

Рисунок 3.1 – Структурна схема системи

В Intel SoC (там де ця підсистема називається Intel TXE) як базова модель для ME-контролера використовується SPARC.

Нічого страшного, в Intel про це подбали: у самих останніх платформах (Skylake, чипсети 100 серії, Intel ME 11.x) ME-контролер має архітектуру... x86! У чипсетах тепер живе ще один x86.

Втім, состав компонентів підсистеми Intel ME (з версії 2.0) не змінювався. Це:

1. **ME-контролер** – убудований у чипсет 32-розрядний мікроконтролер типу RISC, що має внутрішні ROM і SRAM.

2. **Регіон ME в SPI флеш-пам'яті**, у якому зберігається розроблена й підписана компанією Intel прошивання ME-контролера (тому, саме Intel ME firmware).

3. **ME UMA** – схована від усіх, крім ME-контролера, область (16 – 32 МБ) в оперативній пам'яті комп'ютера, який він користується в якості runtime-memory для розміщення й запуску прошивання.

4. **Management Engine Interface (MEI)**, раніше відомий як **Host Embedded Controller Interface (HECI)**, – набір регістрів у конфігураційному просторі PCI і область в MMIO, що представляють собою інтерфейс для обміну інформацією з ME-контролером (по суті, єдиний канал зв'язку софта, що виконується на CPU, з підсистемою Intel ME).

5. **Окремий MAC** – контролер канального рівня, що надає ME-контролеру out-of-band доступ до загального фізичного мережного інтерфейсу для віддаленого адміністрування комп'ютерною системою.

6. **Деякі модулі в BIOS**, відповідальні за ініціалізацію платформи й, що повідомляють про результати своєї роботи ME-контролеру через MEI.

У випадку наявності шильдика Intel vPro, до складу підсистеми Intel ME додатково входить BIOS-модуль ME **BIOS Extenstion (MEBx)**, що надає графічний інтерфейс (показаний вище), а також здійснююче включення й конфігурування AMT через MEI.

Таким чином, у нас є середовище виконання ring -3 (так її умовно називають) – 1 штука. Її привілейованість обумовлює здатностями, якими наділений ME-контролер (про їх написане вище), а скритність – повною відсутністю можливості контролювати програмними (і навіть апаратними, в production-версіях плат) засобами.

Архітектура ME-контролера

Усередині ME-контролера, крім мікропроцесора ARC / SPARC / x86:

– ME ROM – енергонезалежна неперезаписувати^ся память, що, у якій зберігається стартовий код ME-контролера;

– ME SRAM – оперативна пам'ять використовувана ME-контролером при неприступності ME UMA, наприклад, на ранніх етапах роботи;

– кеш коду й кеш даних, для підвищення продуктивності при роботі з пам'яттю;

– C-Link (Controller Link) – шина, що дозволяє ME-контролеру взаємодіяти з периферійним апаратним забезпеченням у режимах S5 (System shutdown) / S3 (Sleep mode);

Різні апаратні блоки:

– високоточний таймер і WDT;

– контролер переривань;

– контролери пам'яті й DMA;

– інтерфейс HECI / MEI;

– RNG, акселлератор криптографічних функцій і функцій стиску.

Самий час розібратися в тому, як від модифікацій захищений код, що управляє всім цим багатством.

Прошивання Intel ME

Intel ME firmware, залежно від наповнення, розрізняють двох типів:

– 1.5 МБ, урізані версії;

– 5 МБ, повні версії.

Тип прошивання визначає состав прикладних модулів, у яких реалізовані певні технології (наприклад, AMT, IPT і т.д.). Хоча є й базова частина, однакова для різних прошивань:

– Bring Up, перший запускати^ся модуль, що, із прошивання.

– Kernel, ядро OCPB Thread.

– Деякі драйвери й служби.

В SPI флеш-пам'яті є кілька регіонів:

– Flash Descriptors, у якому зберігаються покажчики на всі інші регіони, а також read / write привілею для користувачів цієї пам'яті. Відзначимо, що звичайно цими дескрипторами забороняється перезапис ME регіону всім, за винятком самого ME-контролера;

– Gb (Gigabit Ethernet);

– ME, тут зберігається прошивання ME-контролера;

– BIOS;

– 3PDS (Third Party Data Storage), опціональний регіон.

Тепер глянемо на сам регіон ME. Це Flash Partition Table (FPT) – таблиця розділів ME firmware. У ній зберігаються покажчики на різного типу (код, дані, віртуальна область, ...) розділи і їхні параметри. Цілісність цієї таблиці контролюється одним байтом чексумми по зсуву 1Bh.

Нас цікавлять executable-розділи, тобто ті, що зберігають здійсненний код. Їх звичайно трохи, розглянемо один з них.

На початку кодового розділу розташовується маніфест, що складається із заголовка (зі службовими даними й ЕЦП) і таблиці модулів.

У наведеному дампі можна побачити 2048-бітний відкритий RSA ключ (модуль за зсувом 80h відносно початку розділу й експонента за зсувом 180h). Далі треба 256 байт сигнатури.

Своїм закритим ключем компанія Intel підписує частина заголовка маніфесту й таблицю модулів, прикладаючи отриманий підпис і відкритий ключ для перевірки.

А от і фрагмент таблиці модулів розглянутого розділу. Ця таблиця містить заголовки модулів, де зазначені деякі параметри й хеш-сума SHA256 (за зсувом 14h усередині заголовка).

Згенерувати власну пару ключів RSA-2048 і підписати ними розділ не вийде через те, що цілісність прикладеного відкритого ключа перевіряється стартовим кодом в ME ROM, у якому зберігається хеш-сума SHA256 відкритого ключа компанії Intel.

Цього цілком достатньо для захисту прошивання від підроблення. Програмно перезаписати ME регіон SPI флеш-пам'яті не можна (пам'ятаємо про дозволи в Flash Descriptors), апаратні засоби, звичайно дозволять обійти це обмеження, але контроль дійсності не виключити.

Подивимося убік захисту від бінарних уразливостей.

Ми побачили, що весь здійсненний код ME firmware розбитий на модулі різного призначення.

В ME-контролера є два режими роботи: привілейовані й користувальницький (аналоги kernel mode і user mode для CPU). Привілейований режим відрізняє, насамперед, можливість доступу до апаратних ресурсів і можливість обігу по адресах поза відведеним цим модулем діапазону пам'яті.

Кожний модуль запускається й працює в заданому (у заголовку цього модуля) режимі.

Распарсив весь ME регіон можна побачити, що привілейований режим використовується ядром ОСРВ і деякими драверами. Службам і прикладними модулям, як і покладено, приділяється тільки користувальницький режим.

Ми показали, що підсистема Intel ME є невід'ємною частиною архітектури сучасних комп'ютерних платформ (на основі чипсетів / SoC Intel). Очевидно, що її компрометація надає потенційному зловмисникові безмежні можливості контролю над платформою: доступ до всього вмісту оперативної пам'яті (системна пам'ять, пам'ять гіпервізора, SMRAM, ACRAM, виділювана пам'ять для графічного ядра – GFX UMA), out-of-band доступ до мережного інтерфейсу

(моніторинг усього мережного трафіку), віддалений контроль як частина штатної функціональності AMT, перезапис будь-яких регіонів SPI флеш-пам’яті. Бонусом до цього є повна відсутність можливостей виявлення.

Це є вагомою причиною для наявності в Intel ME серйозного захисту. Ми вважаємо, що вендори будь-якого мережного встаткування, що вбудовується, повинні прагнути до описаної моделі безпеки. Її характеризують наступні принципи:

– заборона на використання дефолтного пароля, примус до установки сильного пароля (відповідного певним вимогам);

– використання функцій шифрування в мережних протоколах;

– контроль цілісності й дійсності всього здійсненного коду прошивання;

– механізми захисту від експлуатації бінарних уразливостей.

Заздалегідь прокоментую можливі заклики використовувати комп'ютерні платформи на основі CPU і чипсетів від AMD: у них є дуже схожа технологія, називається Platform Security Processor (PSP). Представлена не дуже давно, в 2013 році.

3.3 Розробка функціональної схеми

Функціонально розроблене програмне забезпечення складається з наступних блоків:

– Блок авторизації, автентифікації та захисту інформації.

– Блокування клавіатури.

– Блокування миші.

– Керування службами.

– Перегляд в повноекранному режимі.

– Блок спостереження.

– Блок управління.

– Блок обмежень прав користувача на Host.

Рисунок 3.3 – Функціональна схема розробленої системи

– Динамічне налаштування розмірів буферу даних, які передаються.

– Стиск даних.

– Планувальник задач.

– Чат.

– Перегляд в віконному режимі.

– Робота з менеджером файлів.

– Робота з журналом подій.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

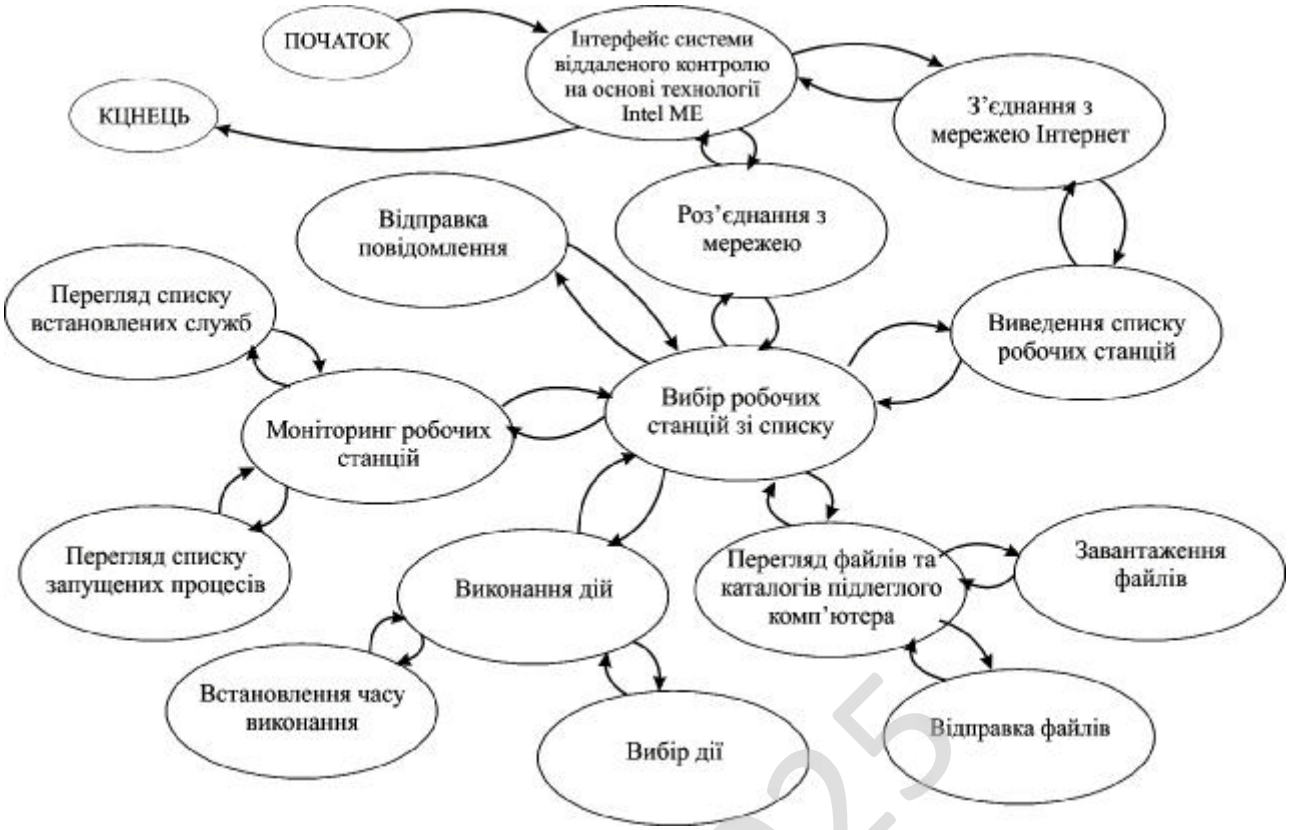

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (ІС), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель

багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що приймально-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості. Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогнозуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу.

Переваги:

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого

проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПО – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

– У модель не входять дії, спрямовані на аналіз ризиків.

– Деякий результат можна отримати тільки при досягненні низу букви V.

Під час роботи над магістерською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято

позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка

потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи віддаленого контролю на основі технології Intel ME, модулю обробки помилок програми і основному модулю.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

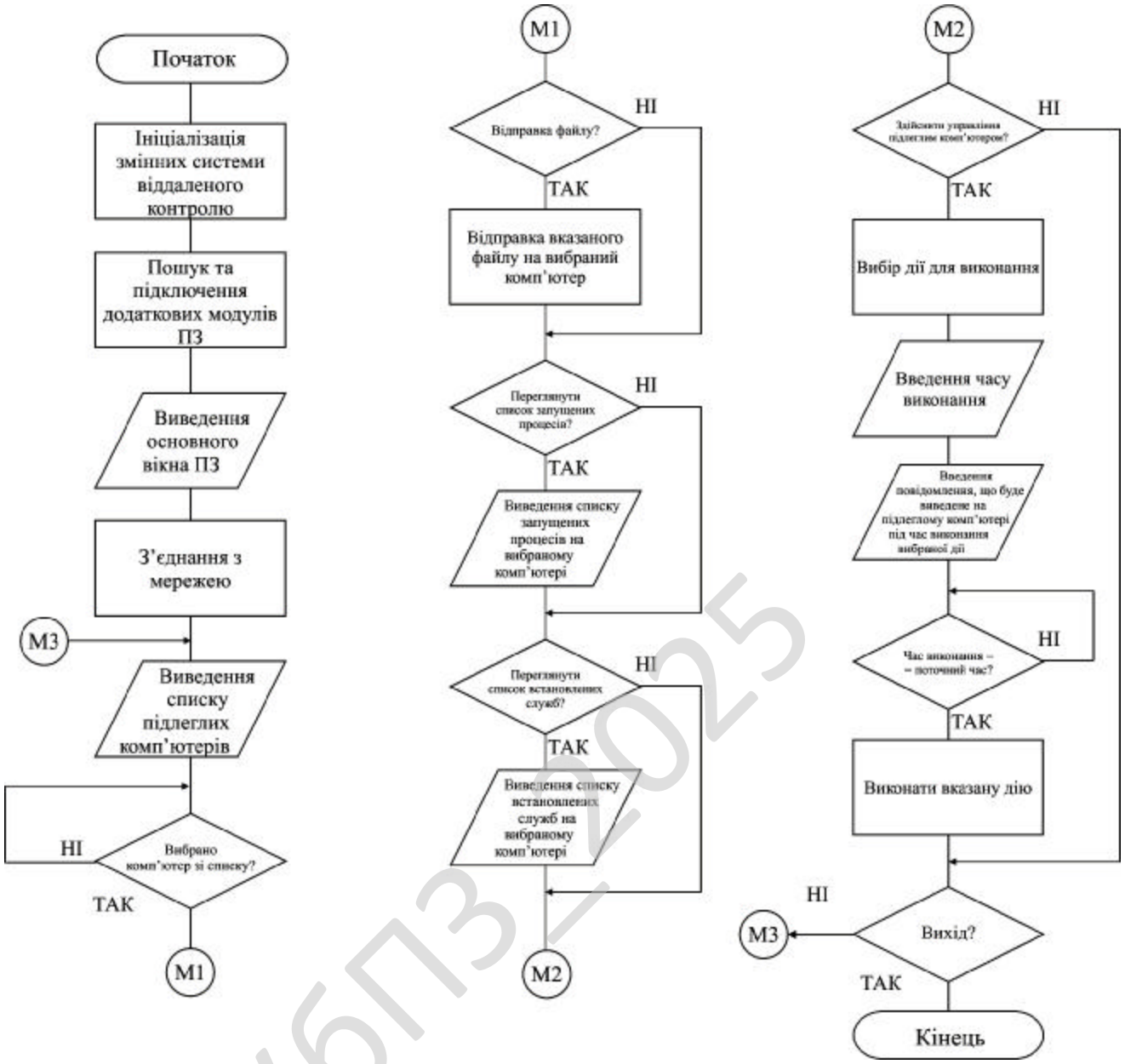

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий

стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

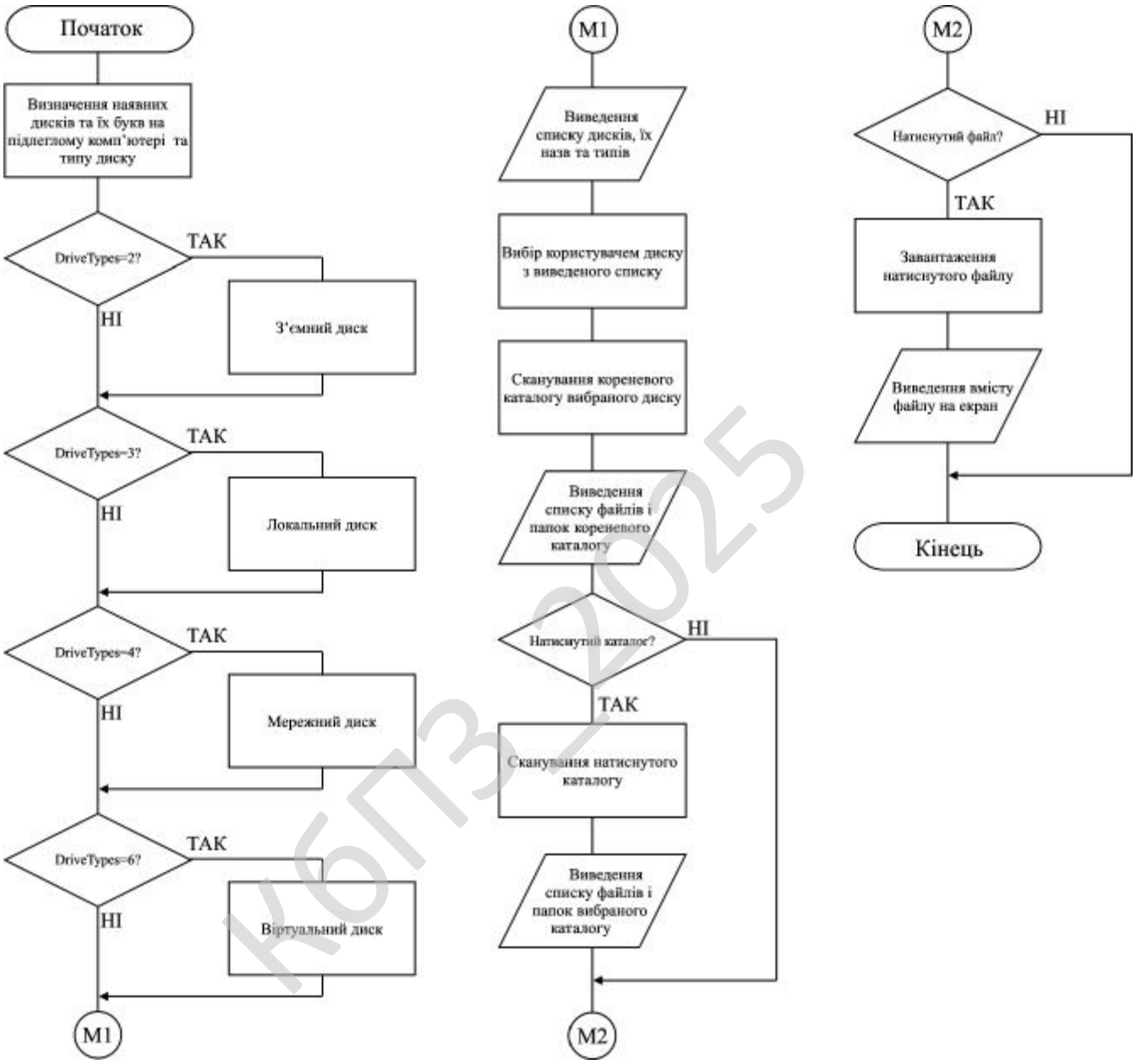

Рисунок 4.2 – Блок-схема роботи підпрограми

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації. Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

Також при розробці магістерської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати

діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та

об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем. Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій.

Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа і-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою: оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний .

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3]. Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].

Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Висновок

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи віддаленого контролю на основі технології Intel ME яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: З’єднання з мережою; Управління комп’ютером ; Параметри; Довідка.

– Функції представлені у графічному вигляді (іконки).

– Розділу обрання групи.

– Розділу виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

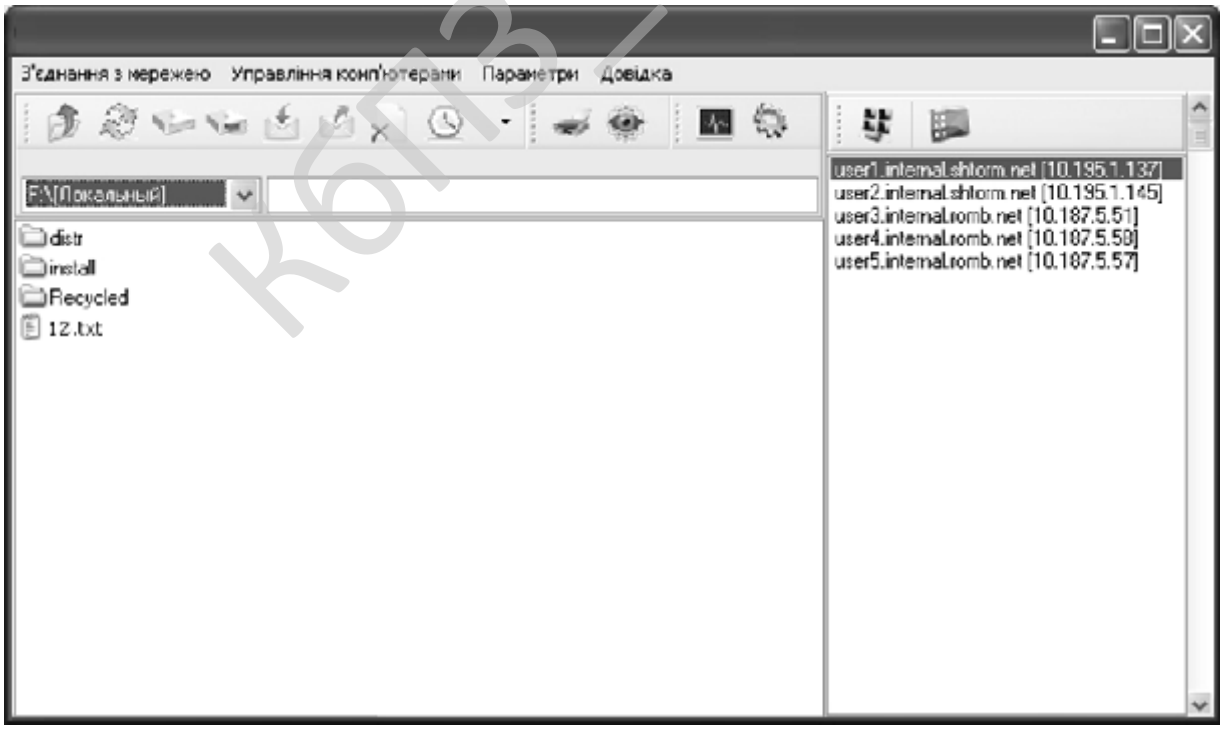

Рисунок 5.1 – Головне вікно ПЗ

Intel Management Engine (ME) – підсистема, що убудована в усі сучасні комп'ютерні платформи (десктопи, лептопи, сервери, планшети) із чипсетами компанії Intel. Ця технологія багатьма сприймається як апаратна «закладка», і на тобто причини. Досить сказати, що Intel ME є єдиним середовищем виконання, що:

– працює навіть тоді, коли комп'ютер виключений (але електроживлення подається);

– має доступ до всього вмісту оперативної пам'яті комп'ютера;

– має позаполосний доступ до мережного інтерфейсу.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем.

Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

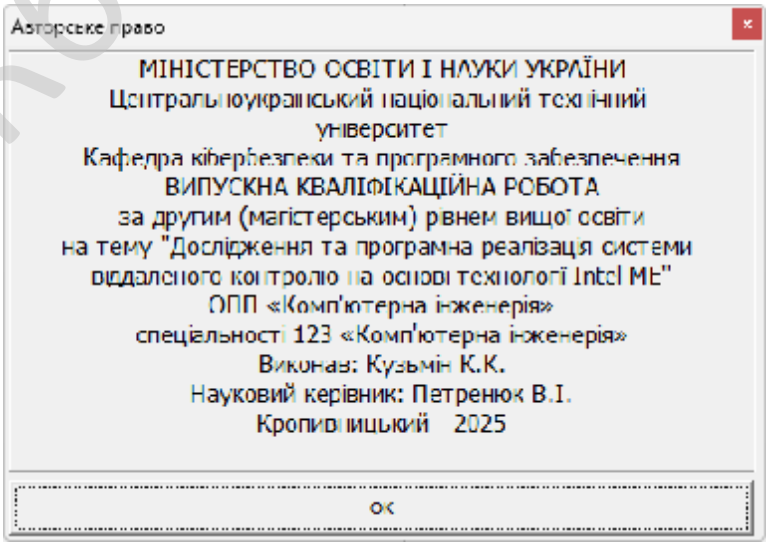

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих

процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт. Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віддаленого контролю на основі технології Intel ME.

Метою розробки є дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME.

Об'єктом дослідження є процес віддаленого контролю на основі технології Intel ME.

Предметом дослідження є методи віддаленого контролю на основі технології Intel ME.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод віддаленого контролю на основі технології Intel ME.

– Розроблено вітчизняний продукт віддаленого контролю на основі технології Intel ME, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження системи віддаленого контролю на базі Intel Management Engine насамперед зацікавлять компанії, які мають велику кількість робочих станцій і прагнуть зменшити витрати на їх обслуговування. Це можуть бути банківські установи, виробничі підприємства, торговельні мережі чи ІТ-компанії, де безперервність роботи комп'ютерів прямо впливає на продуктивність бізнесу. Для таких структур впровадження технології Intel ME відкриває можливість дистанційно оновлювати системи, усувати збої та контролювати стан обладнання без фізичної присутності спеціалістів.

Також ця розробка є актуальною для державних і освітніх установ, де ІТ-інфраструктура розподілена між різними філіями або корпусами. Використання технології Intel ME дозволяє централізовано контролювати комп'ютерні ресурси, що підвищує ефективність управління та знижує ризики технічних збоїв у критичних службах. Результати дослідження можуть бути цікавими також компаніям, що надають послуги технічної підтримки – аутсорсинговим ІТ-компаніям. Використання рішень на базі Intel ME у їх роботі допоможе значно скоротити кількість виїздів спеціалістів до клієнтів, підвищити швидкість реагування на звернення і, відповідно, конкурентоспроможність їхніх послуг.

Крім того, система може стати цінним інструментом для навчальних закладів технічного профілю, де готують фахівців у сфері адміністрування, інформаційної безпеки чи комп'ютерних технологій. Її можна застосовувати як навчальний кейс для демонстрації принципів функціонування апаратно-програмних засобів віддаленого управління. Таким чином, результати реалізації мають широку аудиторію – від бізнесу й освіти до державного сектору.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для визначення привабливості впровадження системи на основі Intel ME було проведено експертне оцінювання серед представників IT-відділів великих компаній та технічних аналітиків. Кожен експерт оцінював систему за критеріями, такими як зручність адміністрування, рівень безпеки, ефективність діагностики, масштабованість і економічна вигода. Результати оцінювання показали, що середній рівень привабливості продукту склав 8,7 із 10 можливих балів.

Більшість експертів відзначили значну перевагу технології Intel ME у порівнянні з традиційними засобами адміністрування, оскільки вона дозволяє контролювати обладнання навіть при відсутності операційної системи або під час збою живлення. Це суттєво підвищує оперативність IT-служби, особливо в умовах, коли критично важливо швидко відновити роботу пристрою.

Додатковою перевагою, на яку звернули увагу експерти, є можливість інтеграції Intel ME з іншими корпоративними системами управління. Це робить рішення гнучким та адаптивним для різних типів підприємств.

Таким чином, результати експертного оцінювання свідчать, що система на базі Intel ME є перспективним і конкурентним інструментом, який поєднує високу ефективність, технологічність і простоту використання.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості впровадження системи на базі Intel ME доцільно застосувати комбінований підхід, який включає витратний метод і метод оцінки ефективності інвестицій. Витратний метод дозволяє точно розрахувати всі прямі витрати – вартість ліцензійного програмного забезпечення, модернізації

комп’ютерного парку, навчання персоналу та налаштування системи. Цей підхід дає можливість отримати чітке уявлення про початкові інвестиції.

Метод оцінки ефективності інвестицій (ROI) допомагає визначити, наскільки швидко система окупиться за рахунок зменшення витрат на обслуговування та скорочення простоїв. Наприклад, якщо раніше компанія витрачала значні ресурси на виїзди IT-спеціалістів до регіональних офісів, то після впровадження Intel ME ці витрати майже зникають.

Поєднання двох підходів дозволяє отримати не лише загальну картину фінансових витрат, а й оцінити реальний економічний ефект. Такий аналіз робить обґрунтування впровадження більш переконливим для керівництва, оскільки демонструє конкретні цифри та терміни окупності інвестицій.

Таким чином, використання комбінованого методу є найраціональнішим варіантом для оцінки вартості реалізації системи Intel ME, адже він дозволяє побачити як витратну, так і вигідну сторони проєкту.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Підприємство має понад 200 робочих станцій і кілька серверів, розташованих у різних офісах. До впровадження системи IT-підтримка здійснювалася локально – технічні фахівці фізично відвідували робочі місця для діагностики, оновлення BIOS, встановлення системи або усунення несправностей. Такий підхід займав багато часу, призводив до простоїв працівників і збільшував витрати на обслуговування.

Для оптимізації процесів було впроваджено систему віддаленого моніторингу та управління IT-парком на базі технології Intel Management Engine (Intel ME). Ця технологія дозволяє керувати комп’ютером навіть тоді, коли він вимкнений або операційна система не завантажується. Вона забезпечує

віддалений доступ до BIOS, оновлення ПЗ, моніторинг стану системи та дистанційне усунення несправностей. Вхідні дані зафіксовано в таблиці 7.1.

Таблиця 7.1 – Вихідні дані для розрахунку

Показник	**До впровадження**	**Після впровадження**	**Економічний ефект**
Кількість адміністраторів	4	2	–2
Середня зарплата адміністратора	40 000 грн/міс	40 000 грн/міс	—
Витрати на адміністрування (рік)	1 920 000 грн	960 000 грн	–960 000 грн
Середня кількість простоїв через збої (год/рік)	600	150	–450
Втрати від простоїв (1 година = 5 000 грн)	3 000 000 грн	750 000 грн	–2 250 000 грн
Витрати на впровадження Intel ME (обладнання, ліцензії, навчання)	—	1 200 000 грн	—
Щорічні витрати на підтримку системи	—	150 000 грн	—

Розрахунок економічного ефекту демонструє наступне: економія на адміністративних витратах – 960 000 грн/рік, зменшення збитків від простоїв – 2 250 000 грн/рік, сукупний річний економічний ефект – 3 210 000 грн/рік, чистий

економічний ефект – 3 060 000 грн/рік, термін окупності (Payback Period) – 0,39 року (~5 місяців), коефіцієнт рентабельності (ROI) – 255 %.

Додаткові (немонетарні) переваги: безперервність роботи: технічна підтримка може втручатися навіть під час збоїв операційної системи, що скорочує час простоїв, безпека: контроль доступу до BIOS і шифрування каналів з'єднання гарантують захист даних, зручність управління: централізована консоль дозволяє моніторити стан усіх пристроїв у мережі, зниження людського фактору: усунення необхідності фізичного втручання зменшує ризик помилок адміністраторів, екологічність: скорочення виїздів ІТ-персоналу знижує витрати палива та вуглецевий слід.

Таким чином, технологія Intel Management Engine стала не лише засобом економії, а й ключовим елементом цифрової стратегії управління ІТ-інфраструктурою, що підвищила ефективність, безпеку й гнучкість бізнесу.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування проєкту системи Intel ME доцільно почати з демонстрації її реальних можливостей. Найкраще організувати пілотний проєкт для кількох компаній або установ, щоб показати, як віддалене адміністрування дозволяє економити час і ресурси. Такі демонстрації створюють довіру та допомагають клієнтам побачити практичну користь системи.

Далі варто активно співпрацювати з ІТ-дистриб'юторами та інтеграторами, які вже мають досвід роботи з корпоративними клієнтами. Через партнерські програми можна розширити охоплення ринку та зробити впровадження системи більш доступним.

Важливим елементом просування є інформаційна кампанія, орієнтована на ІТ-спеціалістів. Статті, відеоогляди та вебінари допоможуть популяризувати технологію Intel ME, пояснюючи її переваги у зручній, практичній формі.

На завершальному етапі варто створити навчальну платформу для сертифікації ІТ-адміністраторів, які працюватимуть із системою. Це допоможе не лише підвищити рівень довіри до продукту, а й сформує спільноту фахівців, здатних якісно впроваджувати технологію на підприємствах.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Оптимізувати канали збуту можна через комбінацію прямих продажів і партнерських програм. Найефективніше буде налагодити співпрацю з офіційними дистриб'юторами обладнання Intel, які вже мають клієнтську базу корпоративного сегменту. Це забезпечить швидкий вихід продукту на ринок і зменшить витрати на власний маркетинг.

Паралельно варто розвивати онлайн-канали комунікації: сайт, соціальні мережі, галузеві форуми. Створення контенту у форматі відеоінструкцій і коротких кейсів допоможе продемонструвати легкість інтеграції системи у вже наявну ІТ-інфраструктуру.

Ще один напрям – співпраця з освітніми установами, які готують фахівців з адміністрування. Включення технології Intel ME у навчальні програми дозволить підготувати майбутніх спеціалістів, знайомих із продуктом, що підвищить його популярність у майбутньому.

Оптимізація шляхів реалізації передбачає також створення технічної підтримки, доступної онлайн. Це забезпечить оперативне вирішення проблем клієнтів і підвищить довіру до продукту. Завдяки цьому система стане привабливою не лише технологічно, а й сервісно.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовими факторами успіху проєкту є надійність технології, простота її впровадження та реальний економічний ефект для користувачів. Успіх системи

визначається тим, наскільки вона здатна вирішити конкретні проблеми підприємства: зменшити час простоїв, полегшити роботу ІТ-відділу, скоротити витрати на обслуговування техніки.

Важливу роль відіграє стабільність і безпека самої технології. Intel ME має вбудовані засоби захисту, що гарантують безпечний віддалений доступ до комп'ютера, і це є вирішальним чинником для компаній, які працюють із конфіденційними даними.

Не менш важливою складовою успіху є підтримка користувачів після впровадження. Надійна технічна допомога та регулярне оновлення системи забезпечують довіру клієнтів і сприяють формуванню позитивної репутації продукту.

Успіх проєкту також залежить від грамотної комунікації з ринком. Чітке пояснення переваг технології, демонстрація її можливостей і акцент на практичній користі формують довгостроковий інтерес до системи. Саме поєднання технологічної досконалості та орієнтації на потреби користувача є основою успіху Intel ME.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Наявний в даний час в нашій країні комплекс розроблених організаційних заходів та технічних засобів захисту, накопичений передовий досвід роботи ряду обчислювальних центрів показує, що є можливість домогтися значно більших успіхів у справі усунення впливу на працюючих небезпечних і шкідливих виробничих факторів. Проте стан умов праці та його безпеки в ряді обчислювальних центрів (ОЦ) та підприємств ще не задовольняють сучасним вимогам. Оператори ЕОМ, оператори підготовці даних, програмісти та інші працівники ОЦ та підприємств ще стикаються з впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика і інші.

Багато працівників ОЦ та підприємств пов'язані з впливом таких психофізичних факторів, як розумова перенапруга, перенапруження зорових і слухових аналізаторів, монотонність праці, емоційні перевантаження. Вплив зазначених несприятливих факторів призводить до зниження працездатності, викликане розвиваються втомою. Поява і розвиток втоми пов'язане зі змінами, які виникають під час роботи в центральній нервовій системі, з гальмівними процесами в корі головного мозку. Наприклад сильний шум викликає труднощі з розпізнаванням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5 – 12% продуктивність праці. Тривала дія шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30 – 60%.

Медичні обстеження працівників ОЦ та підприємств показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення

слуху. Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Аналіз травматизму серед працівників ВЦ показує, що в основному нещасні випадки відбуваються від впливу фізично небезпечних виробничих факторів при заправці носія інформації на обертовий барабан при знятому кожусі, під час співробітниками невластивих їм робіт. На другому місці випадки, пов'язані з дією електричного струму.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному

і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов

для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – мало насичені відтінки холодного зеленого або блакитного кольорів.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину	Не менше 30
20... 40 $м^3$ на людину	Не менше 20
Більше 40 $м^3$ на людину	Може біти використана природна вентиляція

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [2]

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців ІТ-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців ІТ-індустрії і стандартів підприємств,

центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців ІТ-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців ІТ-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Для більшого розуміння, пропозиції щодо підвищення працездатності ІТ-фахівців, розіб'ємо на декілька категорій:

1. Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням ІТ-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють ІТ-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання, переохолодження ІТ-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці ІТ-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

2. Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність ІТ-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві ІТ-галузі. Тому нами пропонується закупівля тільки меблів, які

пройшли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, таймбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.3 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Розрахункова частина

Система освітлення робочого місця користувача ПК має відповідати наступним вимогам (рис. 8.1).

Рисунок 8.1 – Вимоги до системи освітлення робочого місця користувача ПК

Проведемо розрахунок штучного освітлення за методом коефіцієнту використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4.4 м, висота – 3 м.

У зазначеному приміщенні працює 2 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F = E \cdot S \cdot K \cdot Z / n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк, *E = 300 Лк;*

S – площа освітлюваного приміщення.

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i = S / (h \cdot (A + B)),$$

де: *S* – площа приміщення, *S = 13.2 м²; h* – розрахункова висота підвісу, *h* = 3 *м. (співпадає з в*исотою стелі, оскільки лампи освітлення закріплюються на стелі)*;* *A* – ширина приміщення, *A = 3 м; B* – довжина приміщення, *B* = 4,4 *м.*

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$i=1,1.$$

Знаючи індекс приміщення, за знаходимо *n = 0,46 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [8].* Підставимо всі значення у формулу, визначимо світловий потік: F=14204 Лм.

Для розрахунку будемо використовувати стельові *світлодіодні панелі* Призма-72 6400К, світловий потік яких $F_л$ = 7200 *Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та кількість світильників:

$$N = 14204 / 7200 = 1,9 \text{ шт.}$$

Приймаємо необхідну кількість *світлодіодних світильників* 2 шт.

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи віддаленого контролю на основі технології Intel ME.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів віддаленого контролю на основі технології Intel ME.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем віддаленого контролю на основі технології Intel ME.

– Досліджена система віддаленого контролю на основі технології Intel ME.

– На основі отриманих результатів досліджень створена програмна реалізація системи віддаленого контролю на основі технології Intel ME.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віддаленого контролю на основі технології Intel ME.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MISTY1.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузьмін К.К. Дослідження та програмна реалізація системи віддаленого контролю на основі технології Intel ME // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

2. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи. Підручник / В.Г. Оліфер, Н.А.Оліфер. – [5-е вид.]. – 2016. – 944 с.

3. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.

4. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

5. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

6. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

7. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

8. Ramon Nastase «Computer Networking: The Beginner’s guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

9. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

10. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп’ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». *Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару, присвяченого 125-річчю Національного університету «Запорізька політехніка»* (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.). Запоріжжя: НУ «Запорізька політехніка», 2025. С.82-91.

11. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

12. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

13. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

14. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

15. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

16. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

17. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

18. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

19. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

20. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

21. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

22. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

23. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

24. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

25. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

26. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

27. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

28. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

29. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

30. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

31. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

32. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

33. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

34. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

35. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

36. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

37. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

38. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

39. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

40. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

41. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

42. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

43. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

44. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

45. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

46. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

47. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

48. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

49. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

50. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

51. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.