Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: **«СИСТЕМА ОБМІНУ МИТТЄВИМИ ПОВІДОМЛЕННЯМИ В КОРПОРАТИВНІЙ МЕРЕЖІ»**

Виконав: студент 2 курсу, групи 2КІ-18м
спеціальності
123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)

Карплюк Сергій Володимирович
(прізвище та ініціали)
Керівник доц. каф. ОТ, к.т.н. Савицька Л.А.
(прізвище та ініціали)
Рецензент проф. каф. МБІС д.т.н. Яремчук Ю.Є.
(прізвище та ініціали)

м. Вінниця - 2019 рік

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ В ГАЛУЗІ ОБМІНУ МИТТЄВИМИ ПОВІДОМЛЕННЯМИ В КОРПОРАТИВНІЙ МЕРЕЖІ

1.1 Аналіз даних за характером впливу під час обміну миттєвими повідомленнями в корпоративній мережі

Під час обміну миттєвими повідомленнями в корпоративній мережі за характером впливу на отримувача дані можуть бути спонукальними і констатуючими.

Спонукальні дані доносяться найчастіше в наказовій формі, часто у вигляді прохання чи інструкції, гарний тон – у вигляді поради. Така інформація розрахована на те, щоб стимулювати конкретні дії отримувача і виконує такі функції як [1-3]:

1) активізує поведінку отримувача;

2) забороняє чи обмежує певні дії отримувача, або ж небажані різновиди його чи її діяльності;

3) часто – порушує чи корегує деякі автономні властивості поведінки та діяльності.

Констатуючі дані під час обміну миттєвими повідомленнями в корпоративній мережі виступають у формі повідомлення і передбачають зміну поведінки отримувача не прямо а опосередковано і водночас поступово. Варто сказати, що характер повідомлення може бути різним, але часто буває так, що саме від байдужий тону викладу призводить до включення в сам текст повідомлення для отримувача елементів переконання.

Ефект обміну такими даними під час обміну миттєвими повідомленнями в корпоративній мережі потужно залежить від відповідності мети і змісту повідомлення, також, адекватності тих знакових систем, якими користується комунікатор-джерело та людина, що отримує ці дані. Розглянемо для прикладу гіпотетичне підприємство, спеціалізацією якого є

ремонт і налагодження обчислювальної та комп’ютерної техніки. Важливою умовою є те, що ремонтний відділ знаходиться у одному географічному місці, а в іншому – знаходяться користувачі.

Як досить поширений результат маємо таке: майже про 90 відсотків проблем користувачі сповіщають інженерів телефоном, а враховуючи, такий факт, що нетехнічний користувач не здатний кваліфіковано й однозначноописати технічну проблему, що виникла, то зазвичай така проблема вирішується методом виїзду інженера на місце подій. Якщо відверто, то на практиці більша частина подібних «проблем» може бути досить легко усунена самим користувачем за умов наявності більш спеціалізованого засобу зв’язку, ніж звичайний телефонний.

Аналіз отриманих даних під час обміну миттєвими повідомленнями в корпоративній мережі також показує, що частіше за все, спілкування електронною поштою також взагалі не вирішує цю проблему, оскільки і в цьому випадку також все зводиться до проблеми малокваліфікованого опису задачі і пустих намагань інженера зрозуміти її походження.

У даній роботі, відповідно проведеному аналізу, ставиться мета – створення такого додаткового засобу зв’язку, який б допомагав формалізувати подібні випадки за допомогою механізму обміну миттєвими повідомленнями в корпоративній мережі до такого рівня, аби професіонал-інженер міг ефективно визначати проблему на основі словесної інформації від користувача.

Нині, звісно, існують такі спеціалізовані програмні засоби, що застосовуються під час обміну миттєвими повідомленнями в корпоративній мережі для вирішення подібних проблем і задач і найпоширеніші (в тому числі, в державних організаціях) є:

- Instant Messenger;
- ICQ;

– Outlook Express.

Суттєвим недоліком цих вищенаведених систем командної роботи для обміну миттєвими повідомленнями в корпоративній мережі є їх тотальна орієнтація на повідомлення довільного формату, що й уможливлює малокваліфікований опис проблеми.

1.2 Огляд систем командної роботи для обміну миттєвими повідомленнями в корпоративній мережі

Системи командної роботи для обміну миттєвими повідомленнями в корпоративній мережі або системи уніфікованих комунікацій – це спеціалізоване об’єднання різного роду інформаційних технологій в єдину керовану інформаційну систему з метою швидкого і зручного обміну даними, такими як [4-6]:

1) голос;
2) відео;
3) миттєві повідомлення;
4) web;
5) e-mail;
6) голосова пошта;
7) факс;
8) а також засоби забезпечення можливості командної роботи з документами за допомогою всіх можливих сучасних способів зв’язку.

У даній роботі ми розглядаємо системи уніфікованих комунікацій, як інтегровану систему продуктів і технологічних рішень, які постійно розвивається, постійно задовольняють зростаючі потреби користувача завжди і скрізь бути в мережі інформаційного простору. Відомо, що нині сучасні

пристрої та додатки все більше та більше набувають рис єдиного продукту, що дає можливість легко та інтуїтивно обмінюватися будь-якими даними в будь-який час в будь-якому місці.

Задачі, що автоматично вирішуються за допомогою застосування систем уніфікованої комунікації;

1) Побудова територіально цілком розподілених систем миттєвого обміну даними. У більшості випадків в подібних системах уніфікованих комунікацій використовується мережевий протокол IP, що дає можливість забезпечити ненадійний сервіс обміну даними в будь-якому місці нашої планети, де є підключення до глобальної мережі інтернет.

2) Суттєве скорочення операційних витрат на дорого вартісні телефонні розмови, а також, можливість економити на відрядженнях технічних працівників, не втрачаючи при цьому дорогоцінний час на їх дорогу і водночас забезпечуючи таку ж, а іноді і вищу, результативність ділових переговорів, якби то було під час особистої зустрічі. Така суттєва мінімізація витрат з експлуатації та обслуговування тяжкої кабельної інфраструктури, інформаційних систем передавання інформації, а також, підтримка обчислювальної інфраструктури з метою зниження негативних відгуків клієнтів.

3) Стабільне забезпечення високої доступності даних керівництву компанії з метою оперативного прийняття рішень в екстрених та штатних ситуаціях.

4) Суттєве прискорення бізнесових процесів, що засновані на внутрішніх і зовнішніх комунікаціях за допомогою використання співробітниками своїх особистих пристроїв в корпоративних цілях.

5) Оптимізація постійних витрат на утримання та експлуатацію окремих спеціалізованих систем передавання інформації і підтримки зв’язку. Утримання декількох різних таких систем, як і, до слова, різних служб їх підтримки, ніяк не виправдано як в економічному форматі, так і з точки зору їх керованості чи/або ефективності ведення бізнесу (тоді збільшується час

реакції на вирішення наявної проблеми, є погана доступність даних, наявне «розпорошення» відповідальності технічних служб і т.д., і як логічний наслідок – неробочий стан сервісу).

6) Виправдана необхідність у побудові спеціального захищеного середовища для обміну даними, такими, як:

- голос;
- відео;
- миттєві повідомлення;
- web-конференції і т.п.

7) Запобігання витоку даних, забезпечення контролю зловмисних дій під час використанні персональних пристроїв співробітників в структурі корпоративної мережі.

8) Забезпечення надійної можливості ефективного ділового спілкування з клієнтами та партнерами. Для цього передбачено створення автоматизованих систем надання даних клієнтам без втрати відчуття безпосереднього спілкування. Це породжує також необхідність в системах комунікацій з клієнтом в один клік.

9) Спільне використання єдиного каталогу контактних даних. Це передбачає інтеграцію служб уніфікованих комунікацій з єдиною системою аутентифікації користувачів на підприємстві, скажімо, Microsoft Active Directory. Ефективним теж є використання єдиного номерного плану в умовах територіально досить рознесеної архітектури корпоративної мережі підприємства.

10) Оптимізація обчислювальних та операційних ресурсів, а також, забезпечення відмовостійкості ресурсів. Нині сучасна архітектура систем уніфікованих комунікацій передбачає повну можливість розгортання довільних сервісів в віртуалізованому середовищі, в хмарних дата-центрах, і такий підхід додає гнучкості під час вирішення завдань, що стосуються не тільки інсталювання, обслуговування, а також і ефективного використання процесорних ресурсів, наприклад, економії електроенергії та безпеки.

Наведемо перелік продуктів і рішень, що застосовуються для створення систем уніфікованих комунікацій:

- IP-телефонія;
- відеокоференцзв’язок та командна робота;
- обмін миттєвими повідомленнями;
- телекомунікація з клієнтами;
- web-конференції.

Тепер розглянемо кожну з них на такому рівні детальності, щоб отримати необхідні вихідні дані для постановки задач роботи.

1. IP-телефонія

Одна з підсистем системи уніфікованих комунікацій, що призначена для передавання голосових даних по IP-мережі. IP-телефонія забезпечує об’єднання всіх типів голосового трафіку в єдину мережу з інтелектуальною маршрутизацією викликів. Впровадження на підприємстві технології IP-телефонії вносить ряд можливостей, що дозволяють заощаджувати кошти на дротових телефонних комунікаціях, забезпечують клієнтів та співробітників всіма сучасними функціями зв’язку, причому, в будь-якому місці і в будь-який час. Застосування IP-телефонії також передбачає високий рівень конфіденційності переданих даних, ця технологія надає єдиний номерний план і найголовніше – що дає можливість глибокої інтеграції з бізнес-додатками. Все це в комплексі робить впровадження технології IP-телефонії на підприємстві вигідною інвестицією, а нині це є і необхідністю в сучасних умовах ведення бізнесу. Виробники засобів IP-телефонії: Cisco, Avaya, Unify.

2. Відео-, коференцзв’язок та командна робота

Потужний розвиток сучасних інформаційних систем для відео- та конференцзв’язку передбачає створення не лише ефекту реальної присутності між кінцевими користувачами, а й засобу, що може передавати емоції і рівні сприйняття інформації співрозмовниками, також – дозволяє підключати

учасників через різні типи та види каналів зв’язку, в тому числі, з різних пристроїв за допомогою web, чату, аудіо-дзвінків, можливість транслювати робочий стіл і відкриті документи. Бізнесові та кошторисні переваги використання таких технологій надзвичайно складно переоцінити:

- зручність і ефективність, швидкість і надійність під час проведення довготривалих зустрічей і переговорів в режимі реального часу без необхідності в дорогих і часто, неефективних, відрядженнях;
- необхідно лише створення спеціальних переговорних кімнат з шумоізоляцією для нарад співробітників компанії і, звісно, спілкування з клієнтами;
- можливість проведення віддалених навчань, консультацій, співбесід під час приймання на роботу і т. д.

Виробниками є такі фірми: Cisco, Radvision, Polycom, LifeSize

3. Обмін миттєвими повідомленнями

Такі програмні засоби дозволяють обмінюватися миттєвими повідомленнями в режимі реального часу. До важливих додаткових функцій цих засобів можна віднести:

- передачу файлів між вузлами;
- нагадування та оповіщення (особисті, групові);
- індикація, що вказує на статус присутності користувача в мережі;
- демонстрація робочого столу користувачів.

Виробники таких засобів це: Cisco, Avaya, Unify.

4. Телекомунікація з клієнтами

Тут мається на увазі організація контакт-центрів для якісного ведення як вхідних, так і вихідних кампаній з питань роботи з клієнтами, також створення гарячих ліній підтримки користувачів. Тут мова йде про обробку будь-яких типів даних:

- голосу;
- відео;
- email;
- sms;
- чатів і повідомлень в соціальних мережах, включно з можливістю проведення аналітики в активних ланцюжках спілкувань.

Нині подібні сучасні контакт-центри такого рівня забезпечують ефективне обслуговування звернень клієнтів в режимі реального часу, тобто, живого спілкування і у форматі спеціалізованих автоматичних голосових додатків, які дозволяють винести конакт з клієнтами на зовсім новий рівень. При цьому модульна архітектура контакт-центрів дозволяє компанії досить гнучко і просто нарощувати необхідний функціонал в залежності від потреб бізнесу. Наведемо ключові функції і можливості таких контакт-центрів:

- інтерактивна взаємодія (т.з., IVR);
- розпізнавання мови;
- інтеграція з системами CRM;
- аналітика якості обслуговування клієнта;
- інтелектуальна обробка викликів в чергах даних;
- інтеграція із різними соціальними мережами;
- збереження контексту звернення до клієнта під час зміни засобів комунікацій.

Виробники таких продуктів: Cisco, Avaya, Unify.

5. WEB-конференції

Тут розглядається можливість проведення веб-конференцій з великою кількістю учасників, такі засоби комунікації дозволяють користувачам обмінюватися даними будь-якого типу в рамках глобальної мережі, в тому числі, за допомогою гаджетів та персональних комп’ютерів або ж звичайних

мобільних пристроїв. Веб-конференції по своїй суті найкращим чином є найефективнішими для:

- проведення презентацій нових продуктів;
- віддаленого навчання користувачів;
- спілкування з партнерами і замовниками;
- з метою проведення вебінарів;
- для проведення вузькоспеціалізованих завдань, на зразок, віддаленої технічної підтримки;
- також, це можливість створення персональних конференц-кімнат з використанням окремих пристроїв відео- та конференцзв'язку.

Виробники таких засобів: Cisco, Avaya, Unify.

1.3 Спеціалізовані служби обміну миттєвими повідомленнями

Спеціалізована служба для передавання миттєвих повідомлень (Instant Messaging Service) – це така служба, що створена з метою забезпечення передавання повідомлень засобами мережі в режимі реального часу, іншими словами, є службою інтерактивного спілкування користувачів [7-9].

Основна фішка такої служби полягає в тому, що окремому користувачеві надають певний ID – унікальний номер. Цей номер (для розуміння – застосуємо аналогію з телефонним номером) власне і ідентифікує окрему особу в мережі. Отримавши номер, тепер користувач за допомогою спеціалізованої програми підключається до мережі сервісу передавання миттєвих повідомлень. Спеціалізований сервер служби фіксує унікальний номер користувача, і тепер сигнал про його підключення буде автоматично надіслано іншим користувачам системи, які вже заздалегідь занесли цю людину до переліку своїх контактів. Таким чином, про факт появи людини «в мережі» користувачів, що внесені до списку контактів одразу стає

відомо у середовищі програми, і тепер кожен знає, чи під'єднаний його співрозмовник і чи готовий він з ним спілкуватися.

Складові спеціалізованої служби обміну миттєвими повідомленнями є такими:

– система ідентифікації користувачів – спеціалізована служба, призначена здійснювати реєстрацію користувачів служби;

– система обліку стану клієнтів – спеціалізована система, що призначена фіксувати повідомлення про стан (статус) клієнтських програм зареєстрованих користувачів, як то: підключений до мережі, не підключений до мережі, відійшов тощо;

– система доставки повідомлень – спеціалізована система, призначена для надсилання миттєвих повідомлень від користувача одному чи кільком адресатам (групова доставка).

Більшість таких чи подібних служб обміну миттєвими повідомленнями побудовано за технологією клієнт-сервер. Всі взаємопов'язані сервери, служби та підключені до них клієнти – всі разом утворюють т.з., ІМ-мережі.

З метою забезпечення сумісності між різними типами мереж був розроблений спеціальний протокол XMPP (від англ. Extensible Messaging and Presence Protocol – розширюваний протокол для обміну миттєвими повідомленнями та інформацією про присутність користувача), раніше більш відомий як Jabber (від англ. jabber – «базікання»). Наразі частина комунікаційних мереж перейшли саме на його використання.

Ключовими функціями ІМ-месенджера є такі як:

– встановлення зв'язку із сервером служби;

– реєстрація користувача в системі;

– визначення та відображення стану всіх зареєстрованих користувачів у мережі та той час;

– введення, передавання та отримання повідомлень різних типів;

– оповіщення користувачів про отримання нових повідомлень;

– формування та впорядкування наявного списку контактів;

– зручна система нагадувань про дні народження абонентів;

– можливість зберігання історії спілкування з кожним Вашим співрозмовником;

– можливість пошуку нових контактів в мережі;

– можливість організації конференцій;

– функціонал пересилання файлів;

– можливість здійснення дзвінків на стаціонарні та мобільні телефони.

1.4 Огляд конкурентів та аналогів систем обміну миттєвими повідомленнями

Всі існуючі нині програми обміну миттєвими повідомленнями можна поділити на дві великі категорії [10-13]:

– універсальні;

– спеціалізовані.

Універсальні носять уніфіковану назву месенджерів, і призначені для обміну довільними миттєвими повідомленнями у текстовому вигляді. У свою чергу, спеціалізовані ж побудовані на основі певного, наперед визначеного спеціалізованого формату даних, який формує «конверт» (т.з., envelope) і містить в собі лише необхідні користувачеві дані. Нині на ринку існуюпредставлені ть такі універсальні месенджери як:

– ICQ;

– AOL Instant Messenger;

– MSN Messenger;

– Jabber;

– Apple Bonjour.

Всі перелічені вище універсальні месенджери мають майже один і той самий графічний інтерфейс, що має перелік користувачів на основній панелі програмного засобу, і набір ряду функцій, що забезпечують власне обмін миттєвими повідомленнями.

Відомо, що чіткого поділу між серверною та клієнтською складовими в цих програмних засобах немає, оскільки вони одночасно виконують роль і сервера (тобто, приймача повідомлень) і клієнта (тобто, передавача повідомлень).

Розглянемо типове діалогове вікно такого програмного засобу на прикладі програмного засобу для обміну миттєвими повідомленнями – універсального месенджера Jabber (рис. 1.1).

Основними компонентами універсального месенджера Jabber є:

– перелік користувачів месенджера. Зверніть увагу, що в даному варіанті універсального месенджера допускається групування користувачів за певною ознакою, і це використовується в одному з режимів її роботи;

– функція безпосереднього передавання миттєвого повідомлення. І у даному варіанті, що розглядається, таких функцій є дві – 1) у вигляді пересилання довільного повідомлення і 2) у вигляді спеціального чату, або ж загального публічного вікна повідомлень, де компактно відображаються всі повідомлення, які приходять для «групи чату».

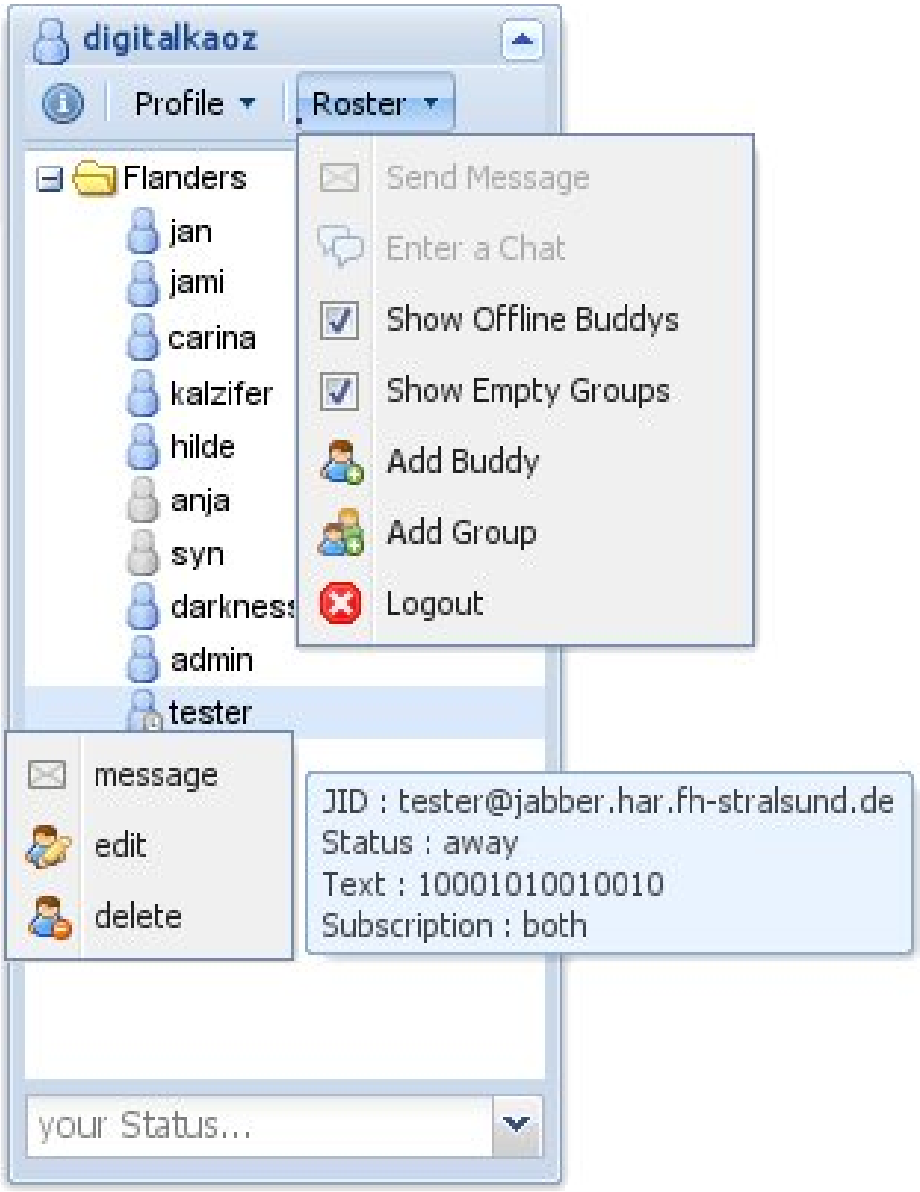

Рисунок 1.1 – Загальний вигляд універсального месенджера Jabber

Оскільки універсальний месенджер Jabber орієнтований на передавання довільних текстових даних, то й відповідно, основне діалогове вікно містить виключно текст, а основним типом даних для нього під час обміну миттєвими повідомленнями є просте текстове повідомлення (рисунок 1.2).

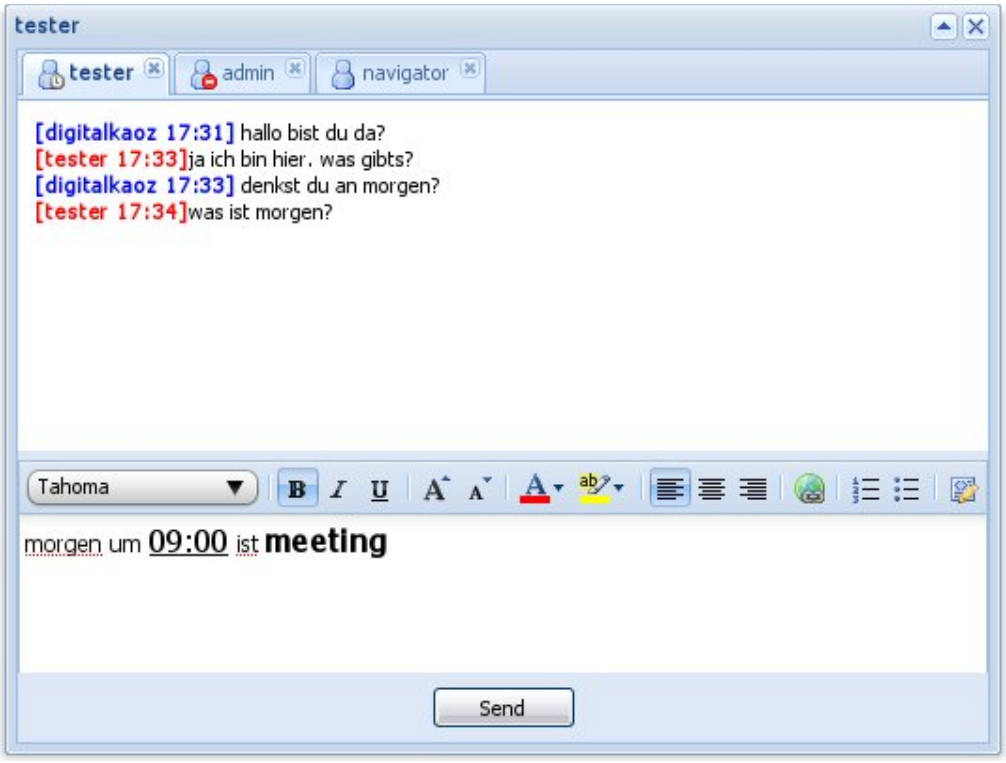

Рисунок 1.2 – Вигляд вікна передавання миттєвого повідомлення

Наведений на рис. 1.2 приклад відображає приклад вікна передавання миттєвого повідомлення, тобто, фрагмент ділового спілкування, а конкретно – призначення наради на 9:00 ранку.

Однак процес обміну миттєвими повідомленнями засобами суто текстових повідомлень довільного контексту не дозволяє, скажімо, контролювати задачу ступеню виконання поставленого завдання, або ж перегляду специфіки того чи іншого завдання, одночасно не вдаючись до пошуку в архіві повідомлень. Тому у галузі обміну миттєвими повідомленнями в корпоративній мережі застосовуються спеціалізовані месенджери, засновані на певному формальному типові даних.

Нині досить складно визначити точно, які ж спеціалізовані месенджери застосовуються найбільше, це пояснюється тим, що через свою вузькопрофільність месенджери мають свою чітко визначену сферу

застосування, а тому практично не можуть бути оцінені за критерієм поширення.

З метою досягнення виконання задач у дипломній роботі, просто підберемо найближчий з ряду месенджерів аналог з точки зору потрібної специфіки, і розглянемо його базову будову.

Отже, виділимо ті месенджери, які призначені для постановлення, контролю та реєстрації завдань чи робіт, і вони діляться на дві великі категорії:

- за календарем;
- за ступінню виконання завдання (роботи).

Серед тих месенджерів, що засновані на принципі календаря, можна розглянути, скажімо My Task Scheduler (рис. 1.3). Такі месенджери, як My Task Scheduler, будуються на базі конкретних часових обмежень, які визначають і перелік завдань, і методи їх контролю.

Суттєвим недоліком месенджерів, що засновані на принципі календаря, є те, що вони зовсім не дозволяють переглянути весь список виставлених завдань чи робіт одразу, а натомість, сортують їх за датою постановки. Крім того, месенджери, що засновані на принципі календаря, не дозволяють визначати ступінь виконання завдань чи робіт.

Натомість інші програми, скажімо, такі, як месенджер Task Coach, дозволяють виконувати категоризування і контроль степеня виконання завдання чи роботи (рис. 1.4).

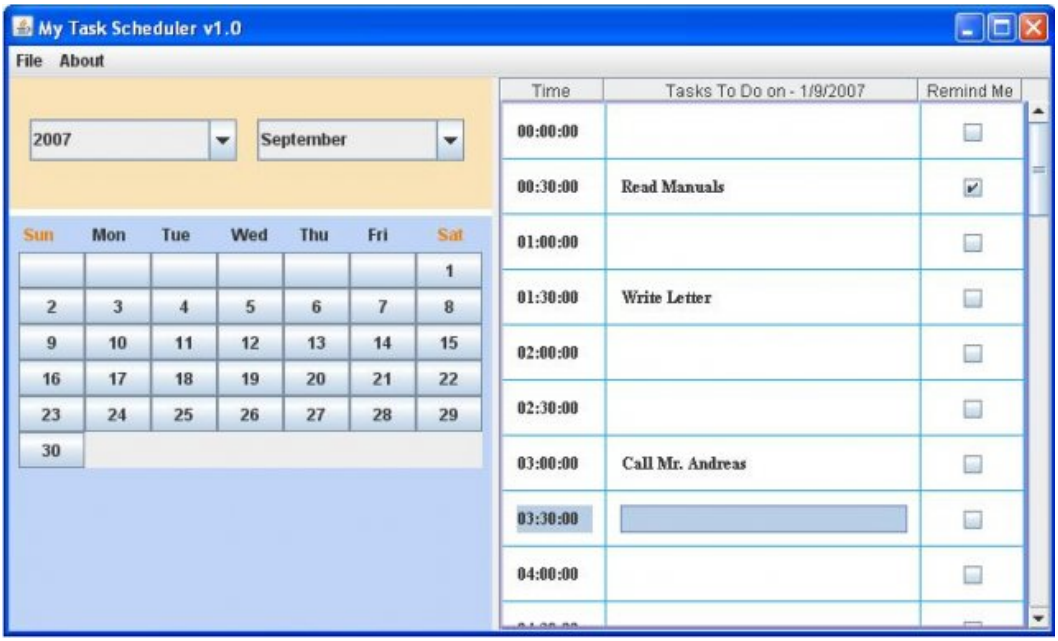

Рисунок 1.3 – Месенджер, що засновані на принципі календаря

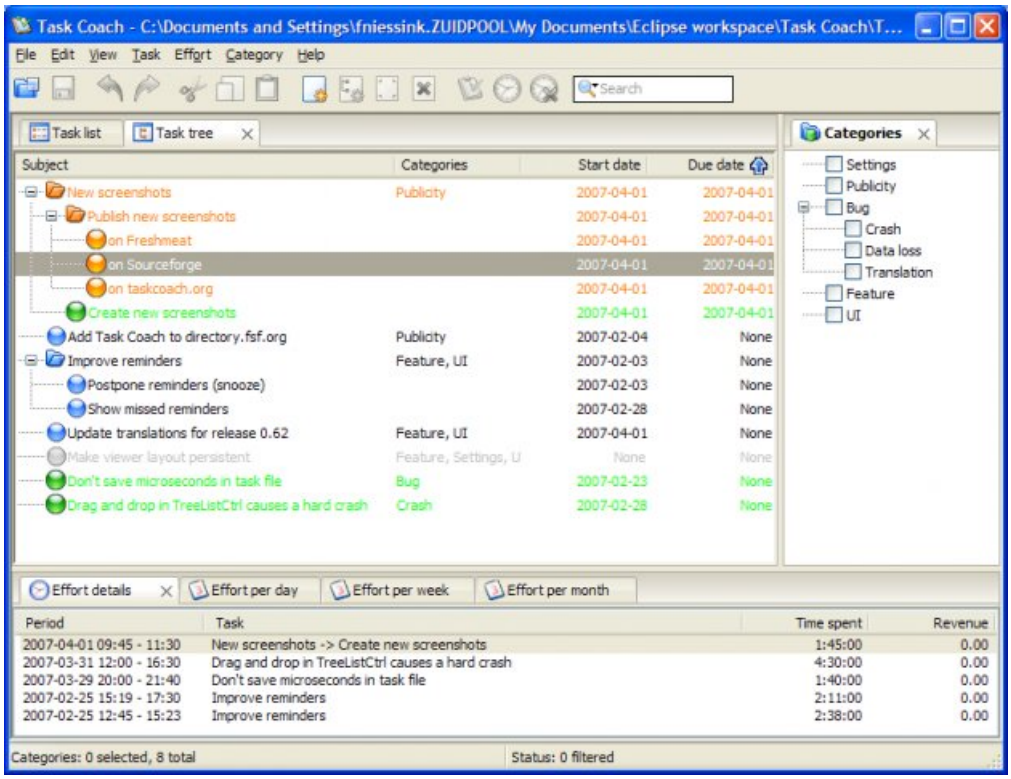

Рисунок 1.4 – Вікно програми Task Coach

Відмітимо, що такі програмні засоби, як Task Coach, мають свій суттєвий недолік – вони надто перевантажені методами роботи із власне процесом контролю виконання завдання чи роботи, і швидш за все, є не засобом обміну миттєвими повідомленнями в корпоративній мережі, а засобом проектного менеджменту [14-17].

Отже, в результаті проведеного аналізу сучасних та присутніх на ринку аналогів програмних засобів для обміну миттєвими повідомленнями, можна зробити висновок про доцільність та необхідність створення такої системи обміну миттєвими повідомленнями в корпоративній мережі, яка би поєднувала у собі простоту інтерфейсу універсального месенджера і функції спеціалізованого месенджера, а також, такого, який би дозволив користувачам ефективно вести обмін миттєвими повідомленнями в корпоративній мережі та одночасно мати функціонал контролю за виконанням спільних задач чи робіт.

1.5 Аналіз вимог до розробки системи обміну миттєвими повідомленнями в корпоративній мережі в галузі командної роботи

Під час аналізу вимог до розробки системи обміну миттєвими повідомленнями в корпоративній мережі в галузі командної роботи, розглянемо спеціалізовані програмні засоби (розглянуті в загальному вище) комунікації в інтернеті – т.з., інтернет-месенджерів [18-20].

Програмний засіб обміну миттєвими повідомленнями в корпоративній мережі і в особистих цілях ICQ є першим і найбільш популярним, та й називається так само, як і однойменний протокол спілкування (крім власне протоколу ICQ є ще інші протоколи, такі як: IRC, AOL, MSN, інші).

Програмний засіб обміну миттєвими повідомленнями ICQ випускається в 2 модифікаціях:

- Lite (полегшена версія);
- Pro (більш технічно просунута версія).

Обидві версії даної програми є вільно розповсюджуваними програмами і є доступними до вільного скачування.

Базова концепція роботи програми ICQ будується на принципі обміну миттєвими повідомленнями між зареєстрованими користувачами мережі ICQ. При цьому, загальний список доступних у даний момент користувачів обмежується внесенням їх у свій список контактів. Звісно, тут захист інформації існує, і з погляду авторизації користувачів (тобто, процесу додавання користувачів у список), то є приховування користувача (режим «невидимий»), коли лише користувачі з особистого списку бачать його або її присутність, до того ж, забезпечується фільтрація повідомлень.

Дистрибутиви програми ICQ для полегшеної і повної її версій розрізняються за розмірами дистрибутивів – це має бути близько 3 мегабайт і близько 10 мегабайт відповідно. Професійна версія програми обміну миттєвими повідомленнями ICQ дозволяє крім відправлення/приймання коротких (усього до 450 символів) повідомлень приймати/пересилати файли необмеженого розміру і типу, а також, відправляти ці повідомлення на адресу електронної пошти, а також, у виді sms-повідомлень на зазначений під час реєстрації номер мобільного телефону користувача, водночас створюючи свою БД користувачів, маючи механізми вести історію повідомлень (це є опціонально), також, підключати різноманітні плагіни для роботи із електронною поштою, стрічками новин і багато чим ще. Полегшена версія програми обміну миттєвими повідомленнями ICQ дозволяє лиш приймати файли і підключати лише самі базові плагіни, вона займає досить мало місця на жорсткому диску комп’ютера користувача.

Основою для процесів спілкування за протоколом ICQ є складання власного контакт-листа користувачів, дані якого доступні на виділеному

сервері, тому робота на чужому комп'ютері зовсім не ускладнюється, як, скажімо, адреса електронної пошти на іншій поштовій скриньці. Процес додавання користувача здійснюється шляхом пошуку переліку користувачів по номерам телефонів, адресам електронної пошти, прізвищам і навіть за реєстраційною інформацією.

Саме в програмі ICQ обміну миттєвими повідомленнями і реалізований механізм редагування в будь-який момент переліку реєстраційної інформації, за винятком ідентифікаційного номера користувача (ID), а також налаштування системи безпеки. У осучасненій версії програми обміну миттєвими повідомленнями ICQ реалізована сумісність з пошуковими системами Google та Rambler.

До прямих переваг системи обміну миттєвими повідомленнями ICQ належить її повна сумісність з БД, що зберігаються на виділеному сервері ICQ, а також, перевагою є простота і невигадливість базового її інтерфейсу як у полегшеній, так і в удосконаленій версіях програми. Додатковою цікавою особливістю програми обміну миттєвими повідомленнями ICQ є можливість посилання/приймання різноманітних вітальних листівок, реалізованих на виділеному сервері ICQ, до того ж листівки надсилаються досить швидко і надійно. Серед переваг є можливість підключення «на ходу» веб-камери і здійснення відеоконференцій, навіть у полегшеній версії програми обміну миттєвими повідомленнями ICQ.

До серйозних недоліків програми обміну миттєвими повідомленнями ICQ варто віднести наявність великої кількості реклами, частину якої, в принципі, звісно, можна деактивувати, проте левова частина все одно залишається активною, а також наявні різні «проломи» в системі захисту самої програми ICQ, що дозволяють встановити на віддалений комп'ютер різні шкідливі програми, скрипти, а також розсилати спам. Програма обміну миттєвими повідомленнями ICQ під час встановлення вбудовується в браузер Internet Explorer, тим самим створює суттєві проблеми для роботи стороні

Low-End-систем, а також є «мішенню» для хакерских атак. Також до всієї системи ICQ відносить такий недолік, що позв'язаний з відновленням паролів до програми (що використовуються для входу на різні системи з багатокористувацьким інтерфейсом) – так вже сталося, що під час реєстрації нового користувача в системі, програма ICQ пропонує відразу два контрольних питання на випадок відновлення забутого пароля, проте відповіді на ці питання, як звичай, добра половина користувачів може забути, і тоді, на виділених ICQ-серверах зберігається велика кількість уже неактивних облікових записів.

Програма обміну миттєвими повідомленнями Miranda IM є другою в переліку програм, що була створена спеціально із урахуванням усіх недоліків програмного засобу ICQ. У Міранді базовий інтерфейс зазнав значних змін, а саме: позбувся від реклами, став використовувати більш надійний механізм Miranda Secure DB і дозволив при досить мінімальному розмірі інсталяційного архіву (а це близько 3 Мб) одержати все те, що входить у професійну версію програмного засобу ICQ. До того ж, з'явилася можливість без обмежень спілкуватися по усім відомим тепер мережевим протоколам, а також значно убезпечити комп'ютер від несанкціонованого доступу з Мережі. Однак, якщо програма обміну миттєвими повідомленнями ICQ призначена для багатокористувацьких систем, то програмний засіб Міранда – для індивідуального спілкування. Програма обміну миттєвими повідомленнями Міранда за замовчуванням зберігає історію повідомлень, контакти в одному файлі БД. Так що під час зміни користувачів програми, до ваших особистих контактів додаються ще і контакти попередніх користувачів програми, а це знижує загальну швидкість роботи програми – власне, як ICQ, так і програму Міранда перевіряють шляхом пінгування виділеного сервера на статус користувачів. Ще однією особливістю роботи програми обміну миттєвими повідомленнями Міранди є її часткова несумісність з іншими сторонніми

клієнтами, наприклад, програм таких, як: ICQ, QI, &RQ, зокрема в процесах передавання файлів.

Серед явних переваг програми обміну миттєвими повідомленнями Міранда варто віднести більш розширені дані про користувачів системи, що включає їх IP-адреси, імена клієнтів, час роботи їх в мережі (значення часу входу-виходу з системи), простий і досить інтуїтивно зрозумілий інтерфейс, свою розширену БД «емоцій», тотальна відсутність реклами, порівняно невеликий обсяг пам'яті, має при цьому більш стабільну роботу і вдосконалені процеси передачі повідомлень довжиною більше за 450 символів. Опціонально в програмі обміну миттєвими повідомленнями Міранда вбудовується велика кількість плагінів, у тому числі і плагін стрічки новин RSS, що дозволяє дізнаватися новини по каналам RSS в автоматичному чи навіть ручному режимі, наявний плагін Weather, що подає прогноз погоди на поточний час і на 5 днів уперед, а також більш вдосконалену у порівнянні з програмою ICQ, має кращу систему графічного оформлення програми (т.з., «скіни»), плагін перевірки пошти за стандартами від GmAIl тощо.

Так само як і програма ICQ, програма Міранда локалізовані під різні мови. Розширені налаштування програм відносяться до практично будь-яких системних подій: написанню якогось тексту, запиту на статус повідомлення, виведенням коротких даних про погоду тощо.

Програма обміну миттєвими повідомленнями Міранда надає можливість зручного перегляду своєї історії повідомлень, у тому числі дає можливість розширеного пошуку по БД повідомлень. Також, особливістю програми обміну миттєвими повідомленнями Міранд є наявність модуля розширеного редагування профілю користувачів системи (що було реалізовано також в програмі ICQ), тому, з метою редагування свої власні дані Вам не доведеться заходити в програму або в Інтернет. Програми Міранда, так само, як і ICQ, виступають також у вигляді «міні-органайзера»,

виконуючи функцію системного нагадування про дні народження користувачів системи.

Так само як і ICQ, і Miranda IM, існують ще два інтернет-месенджери – це QI і &RQ. Розглянемо їх переваги та недоліки.

Переваги програми обміну миттєвими повідомленнями QI;

- зручний користувацький інтерфейс;
- повна відсутність банерів;
- величезна кількість налаштувань;
- велика кількість статус-картинок;
- збереження історії повідомлень в системі;
- має додаткові можливості, що пов’язані із сервісними повідомленнями системи;
- надає можливість перегляду докладних даних в контакт-листі;
- наявний захист від спам-повідомлень.

В останній версії програми обміну миттєвими повідомленнями QI відсутній режим перевірки на «невидимість», оскільки на виділеному сервері ICQ виправили помилку, що дозволяла це здійснювати. У програмі обміну миттєвими повідомленнями QI існує також зручна зміна «скінів», що, звісно приємно.

До мінусів програми обміну миттєвими повідомленнями QI варто віднести такі особливості, як:

- незручна система передавання файлів (через контакт-лист);
- проблеми з процесами кодуванням;
- аватар має бути не більшим за 64 на 64 пікселів.

Так чи інакше, програма обміну миттєвими повідомленнями QI – у порівнянні з програмою Miranda IM – виглядає більш упевнено, проте

вимагає більш продуманих налаштувань, у тому числі і на процеси передавання файлів.

Програмний засіб обміну миттєвими повідомленнями &RQ&RQ зовнішньо дуже схожий на QI, по факту, відрізняючись тільки інтерфейсом. Насправді, це є клон програми QI, із такими функціями:

- функція «невидимості»;
- функція антиспам;
- система керування контактами тощо.

Також тут можна завжди бачити IP-адресу користувача і без дозволу додавати його в список контактів. Як Miranda, програми IM і QI, так і програма обміну миттєвими повідомленнями &RQ є додатком, що не споживає велику кількість трафіку користувача, плюс, потребує значно менших операційних ресурсів, на відміну від програми ICQ.

Програма обміну миттєвими повідомленнями Trillian є також таким програмним продуктом, що надається як shareware. Trillian Pro – це потужний інструмент із потужним інтерфейсом як для професіоналів і комерційних користувачів, так і для тих користувачів, що мають більше однієї мережі, і для тих хто використовує тільки 1 засіб спілкування, або ж навіть для користувачів Інтернету, які шукають ефективний програмний засіб залишатися на зв’язку зі світом. Програма Trillian (Трилистник) надає доступ по всім можливим каналам зв’язку, як то: AI, ICQ, IRC, MSN, Yahoo!

Програма обміну миттєвими повідомленнями Trillian Pro ґрунтується на основі нового програмного ядра і, відтепер, вимагає більшого завантаження ресурсів комп’ютера. З іншого боку, у програмному засобі Trillian об’єднано всі БД по каналах IM, тому немає потреби встановлення значної кількості програм. Як і у вищеописаних програмних засобах, у Trillian реалізовано механізм посилання/приймання миттєвих повідомлень,

файлів застобами повідомлення, тут є розширена функціональність шляхом підключення додаткових плагінів.

Так само як і ICQ, Trillian підтримує Web Cam&Conferences, що дозволяє влаштовувати, у відповідності з відповідними технічними умовами, відеоконференції в мережі Інтернет. Єдиним серйозним мінусом програми є її реєстрація, за принципом ShareWare.

Суттєвим спільним недоліком всіх наведених вище аналогів програм обміну миттєвими повідомленнями є їхня орієнтація на довільний формат повідомлення, що по суті унеможливлює кваліфікований опис проблеми стану вирішення задачі чи роботи в системах колективного виконання задач.

1.6 Висновки за розділом

У даному розділі магістерської дипломної роботи виконано аналіз сучасного стану розвитку систем обміну миттєвими повідомленнями в корпоративній мережі, а саме, проаналізовано дані, що є ключовими під час здійснення обміну миттєвими повідомленнями, виконано огляд та аналіз систем командної роботи та систем обміну миттєвими повідомленнями, розглянуто відповідні спеціалізовані служби, виявлено їх недоліки та сформовано вимоги до програмного засобу, що переслідує досягнення мети магістерської дипломної роботи.

2 СИСТЕМА ОБМІНУ МИТТЄВИМИ ПОВІДОМЛЕННЯМИ В КОРПОРАТИВНІЙ МЕРЕЖІ

2.1 Схема роботи системи обміну миттєвими повідомленнями

В загальному випадку, робота системи обміну миттєвими повідомленнями полягає у обміні даними між двома основними частинами цієї системи [21]:

1) Підсистема обміну миттєвими повідомленнями;
2) Підсистема контролю виконання спільних задач (робіт).

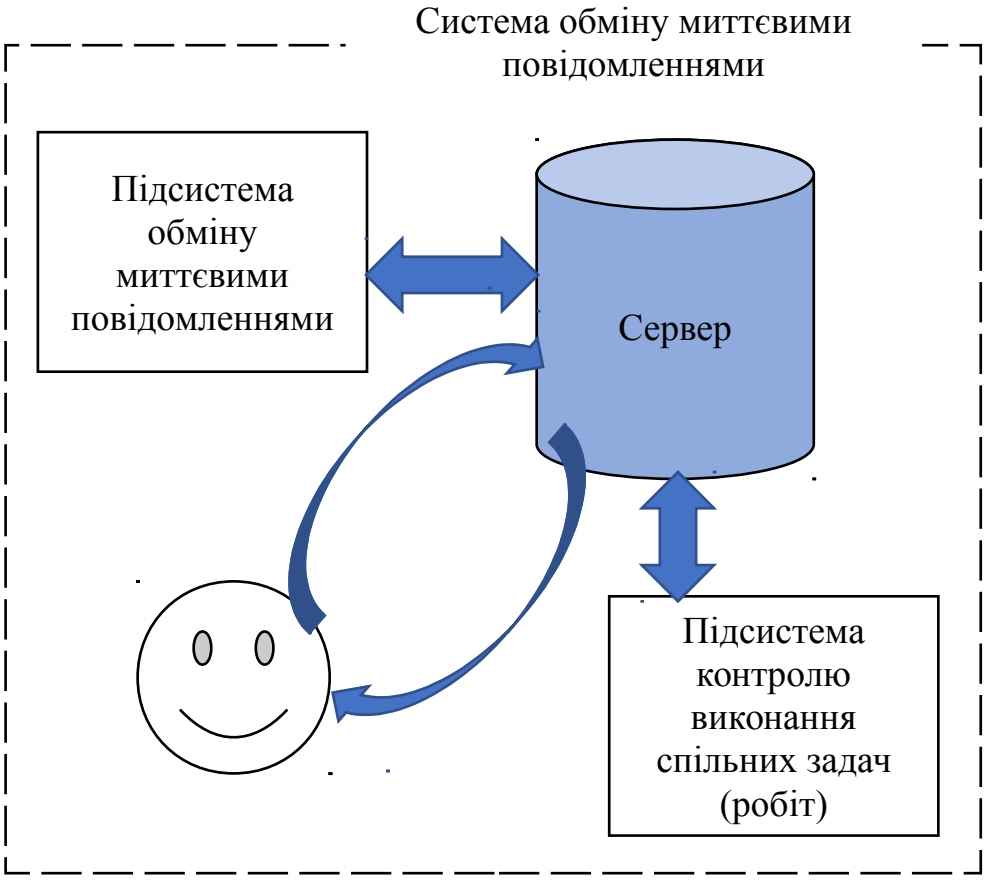

Рисунок 2.1 – Загальна схема роботи системи обміну миттєвими повідомленнями

На схемі 2.1 бачимо таке:

- підсистема обміну миттєвими повідомленнями відповідає власне за обмін повідомленнями між сервером та клієнтом в корпоративній мережі (рис. 2.2.),
- підсистема контролю виконання спільних задач (робіт) відповідає за ходом виконання колективом людей однієї задачі та трекінгом ступеню їх виконання, а також, створюватиме службові повідомлення про стан спільної задачі (рис. 2.3).

Підсистема обміну миттєвими повідомленнями передбачає такі складові у своїй структурі:

1) Інформація про користувача у визначеному типі даних;
2) Перетворення визначеного типу даних у текстові дані (парсинг) для передачі по стекові протоколів в структурі корпоративної мережі – тобто, формування т.з., конверту даних;
3) Передача конверту до модуля передавання даних, за його запитом до стеку протоколів;
4) Сервер за допомогою спеціалізованого модуля отримує конверт;
5) Сервер розпаковує конверт і перетворює його вміст із текстових даних на визначений тип даних (зворотній парсинг);
6) Оновлення списку спільних задач (робіт), чи зміна їх статусу, поява нових, зникнення старих спільних задач (робіт).

Рисунок 2.2 – Підсистема обміну миттєвими повідомленнями

Підсистема контролю виконання спільних задач (робіт) передбачає такі складові у своїй структурі:

1) Постановка спільної задачі (роботи);
2) Отримання конкретного завдання користувачем системи;
3) Виконання конкретного завдання користувачем системи або вказання на ступінь його виконання

4) Зміна (чи оновлення) статусу виконання спільних задач (робіт) та повідомлення про це того, хто виставив задачу;
5) Завершення виконання спільних задач (робіт).
6) Видалення спільних задач (робіт) зі списку.

Рисунок 2.3 – Підсистема контролю виконання спільних задач (робіт)

2.2 Розробка типу даних для процесу обміну повідомленнями

Основою спеціалізованої системи обміну миттєвими повідомленнями є тип (формат) даних, який загортається у спеціальний т.з., «конверт» і виступає базовим суб'єктом обміну миттєвими повідомлення. З метою спрощення роботи користувача системи і побудови схеми цього типу даних було обрано уніфікований формат XML [22-26].

Цей формат одночасно є мовою створення та розробки документів специфічного формату та структури, і носить назву eXtensible Markup Language (XML), що його було прийнято як стандарт для документів певного типу під час консорціуму World Wide Web Consortium (далі W3C), ключовою задачею якого є стандартизація усіх мережевих технологій у 1998 р.

Корисним т.з., побічним ефектом вибору цього формату є його тотальна Internet-орієнтованість, оскільки дедалі більше сайтів за базовий носій інформації застосовують саме його. Крім того, багато серверних додатків також все ширше застосовують XML. Тому використання XML безумовно створює широкі можливості з ефективного подальшого розвитку даного програмного засобу розробки формату даних.

Таким чином, за міжнародним стандартом XML кожен документ має тип і складається з конкретних елементів. Кожен елемент для ідентифікації відокремлюється маркерами початку та кінця:

<ім'я елементу> -- маркер початку
</ім'я елементу> -- маркер кінця

Кожен елемент може містити або вже містить інші елементи в собі, таким чином, може містити навіть текстові дані. Безмовно, допускається і змішаний зміст у форматі – це коли якийсь елемент містить підпорядковані собі піделементи та текст одночасно.

По правді, такої ситуації слід уникати, оскільки це значно утруднює подальший аналіз структури документу. Якщо все ж існує потреба у створенні додаткових даних в окремому елементі, краще скористатись т.з., атрибутами елементу [27-29].

Атрибутом елементу називається певна характеристика, що є притаманна даному елементу. Атрибутів може бути кілька і вони задаються типом документа, або ж безпосередньо користувачем (це відомі так звані вільні атрибути), і несуть додаткові дані та інформаційне навантаження документу. Суто синтаксично вони позначаються таким чином (тут і далі ми будемо використовувати елементи створеного автором формату обміну даними):

<characterictic type="ВидиПовідомлень" />

Наведений вище приклад чітко демонструє так званий принцип «мінімізації». Подібна синтаксична конструкція використовується у тих випадках, коли елемент не має підпорядкованих елементів, і не несе текстових даних у своєму змісті. Тоді атрибут позначається точно так само, як і властивість у звичайних конфігураційних файлах, тобто, за принципом «ім’я=значення», всередині самого маркера початку [30-32].

Кожен елемент документу обов’язково має так званий, «батьківський елемент», в структурі якого він знаходиться. До речі, є лише один елемент, який його не має – це і є власне кореневий елемент. Зазначимо при цьому, що обидва маркери елемента мають знаходитись у структурі одного і того самого елементу, а не в якомусь іншому. Разом усі ці кореневі та складові елементи, з яких складається довільний документ, утворюють зрозумілу деревовидну структуру, на зразок звичайної ієрархічної БД. Такий підхід, до речі, дозволяє програмістам звертатись до структури XML-документів, причому, як до мініатюрних таблиць БД, де роль метаданих відображає визначення типу документу, а роль записів відіграють, власне XML-файли.

Відомо, що довільний XML-документ складається з довільної кількості елементів, кожен з яких або може містити (або вже містить) вкладені елементи, або ж документ може містити текст. У загальному випадку, змістом елемента можуть виступати і інші типи різних даних, але їх розпізнання стає задачею виключно програміста (при цьому, зазначимо таке: автоматизований контроль інформації в загальному випадку не є можливим). Такі принципи та ідея покладені в основу визначення типу документу в загальному сенсі (рис. 2.4.).

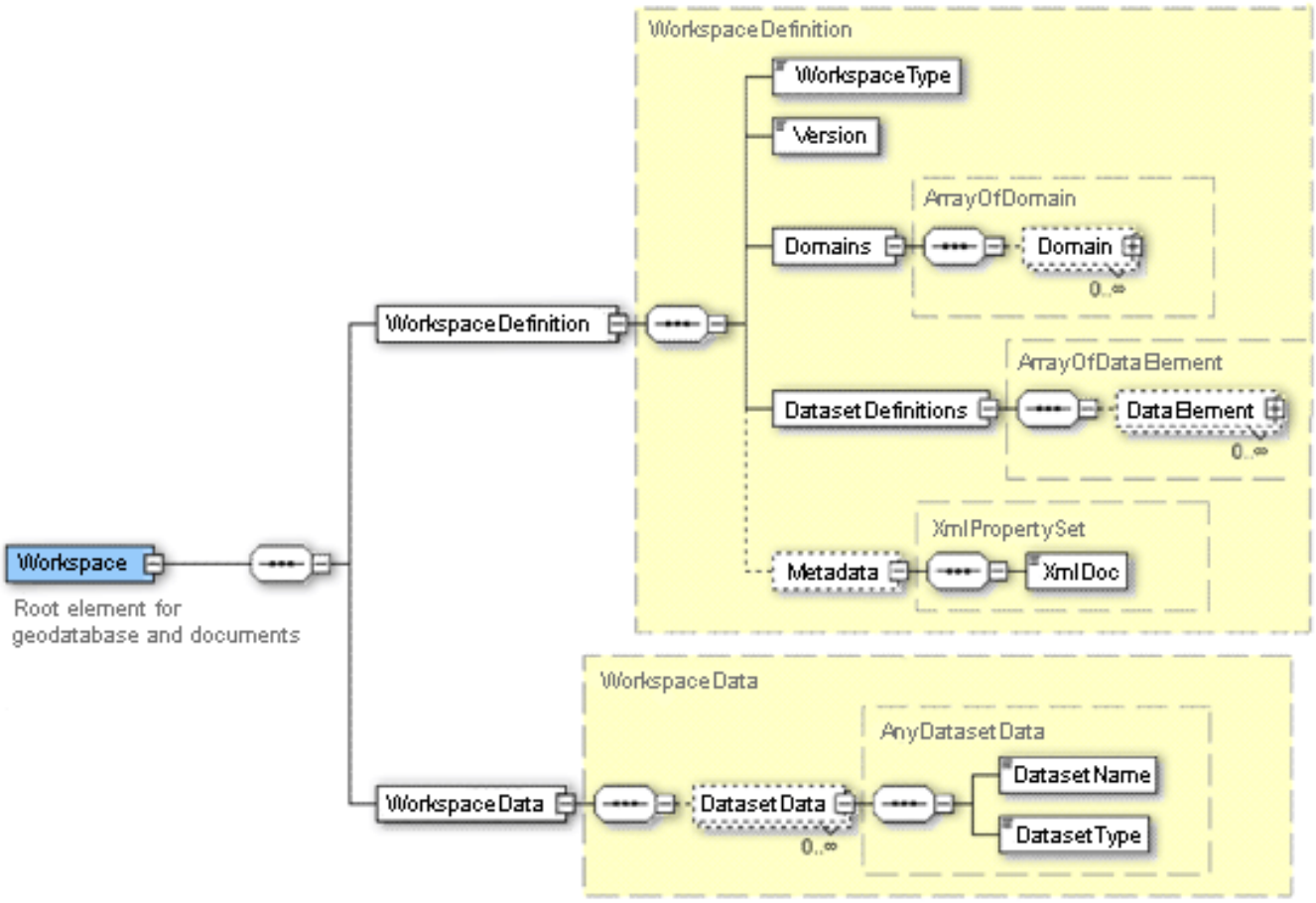

Рисунок 2.4 - Схема документу повідомлення за XML

За таким самим стандартом кожному такому чи кожному розробленому типу документу відповідає його «схема документа». На час розробки даної

дипломної магістерської роботи стандарт XML Schema вже був загальноприйнятим та рекомендованим консорціуму W3C, але програмні засоби, які застосовують цей спосіб задання семантики XML-документів поки що не з’явився на IT-ринку. Слід відмітити, що, незважаючи на широке поширення систем перевірки валідності документів на основі визначення типу (формату) документу, майбутнє все ж таки саме за XML Schema, оскільки він володіє гнучкістю та сильною типізацією даних.

За допомогою визначень на XML Schema можна легко обмінюватись інформацією між XML-представленням документа та іншими джерелами інформації (скажімо, СКБД). До того ж, відомо, що схеми документу синтаксично є XML-документами, а отже під час застосування даної технології відпадає нагальна потреба в реалізації додаткового синтаксичного аналізатора.

Обравши для системи обміну миттєвими повідомленнями основного типу даних та їх формату, нам слід попередньо визначити основні базові елементи розроблюваного типу даних [33-35].

1) Ім’я користувача, який поставив завдання;
2) Стислий опис завдання (шапка);
3) Розгорнутий опис завдання;
4) Відсоток виконання.

Цього необхідно і достатньо для забезпечення ефективної віддаленої постановки задачі і контролю її виконання. Відповідно схема матиме вигляд, як показано на рис. 2.5.

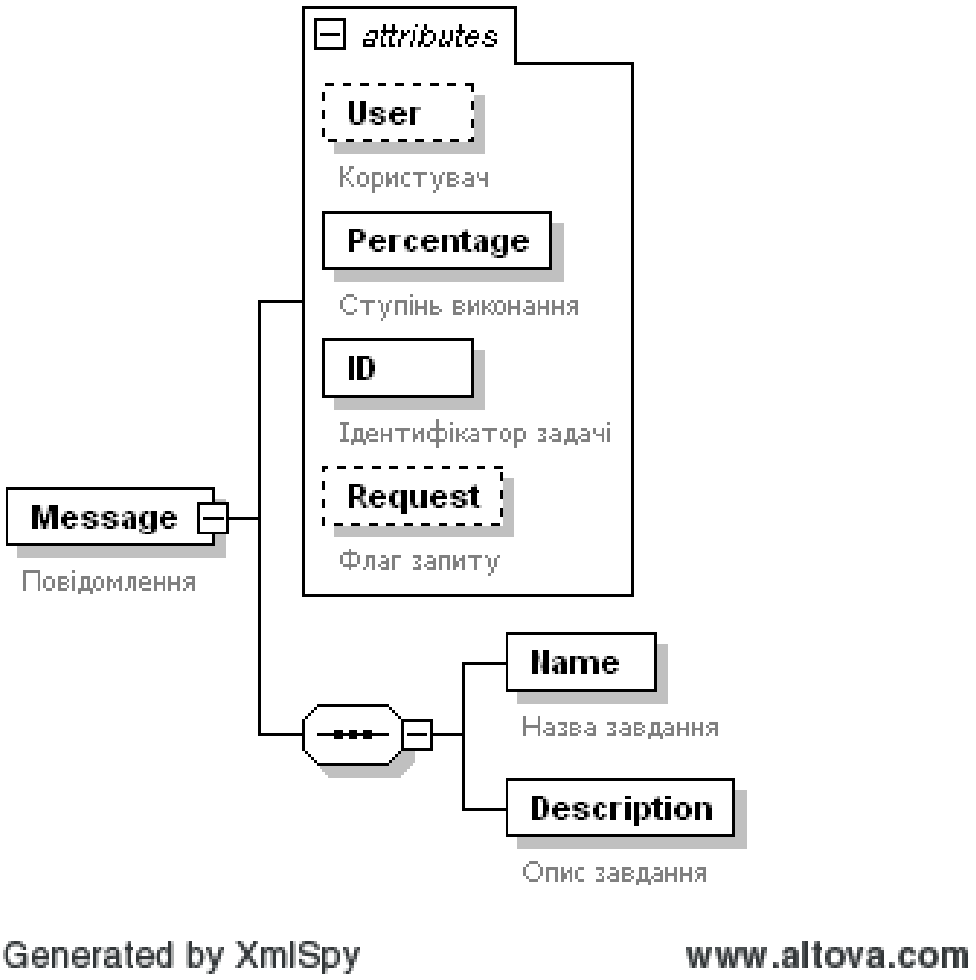

Рисунок 2.5 - Схема документу повідомлення для системи обміну миттєвими повідомленнями

2.3 Модель обміну миттєвими повідомленнями в корпоративній мережі

В даному окремому випадку модель обміну миттєвими повідомленнями для системи обміну миттєвими повідомленнями має бути двосторонньою. Це означає, що за допомогою одного і того самого формату чи типу даних мають передаватись дані про таке:

- перебіг виконання завдання;
- про постановку завдання.

Така схема для систем, де відбувається передавання будь-яких даних, називається повнодуплексною.

Розроблена в попередньому підрозділі схема документу для системи обміну миттєвими повідомленнями містить два сервісних атрибути, які й будуть використані для організації дуплексного зв’язку:

- Атрибут Percentage;
- Атрибут Request.

Перший з цих атрибутів, Percentage – показує на величину поточого відсотку виконання спільної задачі чи роботи, і справедливо є цілочисельним типом даних. Другий атрибут, Request, тут має булевський тип, і позначає (логічними значеннями «0» чи «1»), чи не є дане повідомлення постановкою спільної задачі чи роботи. Відповідно до цього модель обміну миттєвими повідомленнями, що розроблена спеціально для системи обміну миттєвими повідомленнями матиме такий вигляд (рис. 2.6).

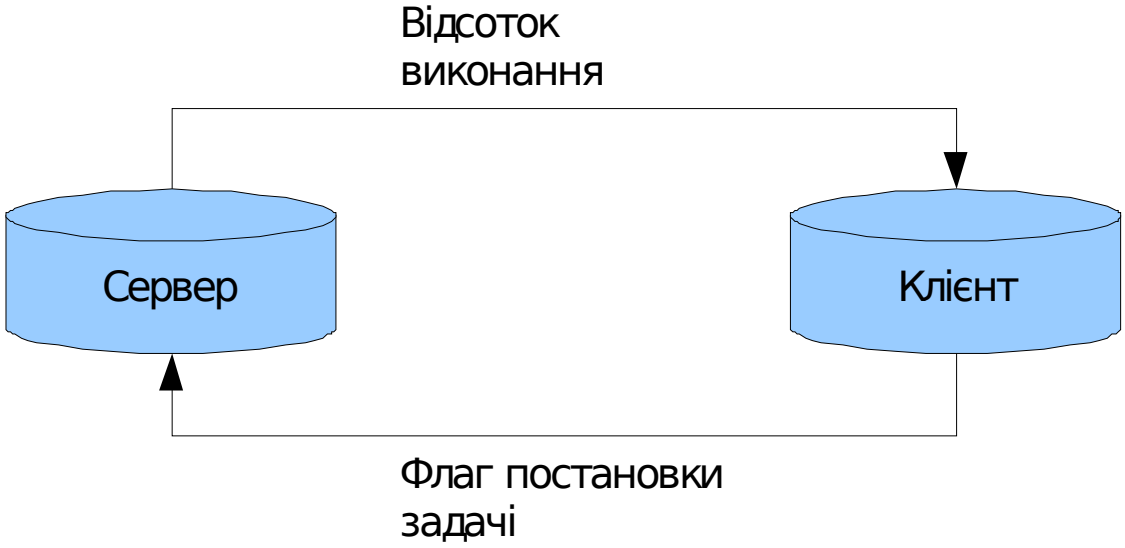

Рисунок 2.6 - Модель обміну миттєвими повідомленнями

Під час використання запропонованої спрощеної моделі обміну миттєвими повідомленнями необхідно використати певну ознаку, таку, за якою можна одне завдання відрізнити від будь-якого іншого завдання. Ще один спеціальний атрибут може стати саме такою ознакою, за якою уніціфкація може бути введена до схеми документа, і має властивості ID (ідентифікатор). Використовуючи цей ID, програміст може ефективно будувати повний цикл обміну повідомленнями для виконанню певної спільної задачі чи роботи.

2.4 Алгоритми обміну миттєвими повідомленнями в корпоративній мережі

Обмін інформацією для розроблюваної системи обміну миттєвими повідомленнями складатиметься із двох частин. Перша частина відповідатиме безпосередньо за обмін миттєвими повідомленнями, а друга, в свою чергу, відображатиме власне процес обміну миттєвими повідомленнями в корпоративній мережі.

Отож, сформуємо кроки, які необхідні для першої частини системи обміну миттєвими повідомленнями:

– Збір інформації про користувача. Введені користувачем дані в діалогове вікно програмного засобу трансформуються в спеціалізований тип даних, що розроблений автором вище.

– Формування конверта в корпоративній мережі. Тут тип даних перетворюється на текстовий тип даних, а потім відбувається перетворення у форму, яка може бути передана по протоколам TCP/IP або UDP.

– Пересилання конверту в корпоративній мережі. Утворений конверт передається в модуль передавання інформації, який звертається до

системного стеку комунікаційних протоколів і відсилає конверт за вказаною IP-адресою.

– Отримання конверту в корпоративній мережі. Протилежна частина програми (сервер) через окремий модуль, що виконує прослуховування мережевого порту, отримує цей конверт і перетворює його на текстову форму;

– Розпаковка конверту в корпоративній мережі. Тепер з текстового представлення конверт даних перетворюється на новий тип даних.

– Доповнення списку спільних задач чи робіт. Новостворений тип даних передається до інтерфейсу користувача, і або 1) утворюється нова позиція по спільній задачі, або 2) вносяться зміни в існуючу спільну задачу чи роботу.

Також, сформуємо кроки, які необхідні для другої частини системи обміну миттєвими повідомленнями:

– Первинна постановка спільної задачі. Користувач пише повідомлення (запит) на постановку завдання і відсилає його на серверну сторону.

– Отримання спільної задачі. Серверна частина на своїй стороні приймає повідомлення про спільну роботу і реєструє ії в загальному списку задач.

– Виконання спільної задачі. Відповідальний за перебігом виконання завдання працівник редагує ступінь виконання в загальному списку завдань.

– Оновлення статусу спільної задачі. Відповідно тому, як змінюється статус спільних робіт автоматично формується службове повідомлення на запит оновити його стан, і надсилається користувачу, який поставив завдання.

– Завершення виконання спільної задачі. Після того як відсоток виконання спільної задачі чи роботи сягає 100%, тоді завдання можна маркувати як завершене, і видалити його із загального списку.

Відповідні блок-схеми наведені на рисунках 2.7 та 2.8.

Рисунок 2.7 – Алгоритм обміну миттєвими повідомленнями в корпоративній мережі на стороні клієнта

Рисунок 2.8 – Алгоритм обміну миттєвими повідомленнями в корпоративній мережі на стороні сервера

2.5 Алгоритм командного контролю виконання задач

Алгоритм командного контролю виконання задач зображений на рисунку 2.9 і передбачає відображення стану готовності спільної задачі, що виконується учасниками корпоративної мережі.

Рисунок 2.9 – Алгоритм командного контролю виконання задач

Алгоритм командного контролю виконання задач передбачає такий перебіг дій: постановка спільної задачі (роботи); отримання конкретного завдання і виконання його користувачем системи (або вказання на ступінь його виконання); зміна (чи оновлення) статусу виконання спільних задач та формування повідомлення про це для того, хто виставив дану задачу; завершення виконання спільних задач і видалення їх зі списку.

2.6 Програмна реалізація системи обміну миттєвими повідомленнями

Основа для серверної частини програмного засобу, що реалізовує систему обміну миттєвими повідомленнями, взята з вільнопоширюваного ресурсу і модифікована так, що за своїми якостями є практично універсальною базою для програм подібного класу. Доцільне певне доопрацювання цієї програми в галузь розширення кількості протокольних команд вітається.

Вся клієнтська сторона була написана автором особисто. Програма для реалізації системи обміну миттєвими повідомленнями працює таким чином:

1) користувач запускає за допомогою програмного забезпечення аплет із сервера;
2) користувач вводить свій ідентифікатор в системі;
3) користувач бачить перелік ідентифікаторів тих користувачів, які під’єднані до сервера і є частиною команди;
4) користувач має змогу обмінюватись повідомленнями з іншими учасниками команди;
5) користувач одержує миттєві повідомлення в разі під’єднання або від’єднання користувачів системи.

Програма складається з 4 класів (модулів):

- Server;
- ClientConnection;
- Client;
- ServerConnection.

Перші два заявлених класи відносяться до серверної частини програми, а інші дві – до клієнтської.

2.6.1 Реалізація серверної частини

Розглянемо в даному підрозділі розробку модуля, що реалізовує створений клас Server [36].

Клас Server в загальному випадку є універсальним і нічого не знає про ті повідомлення, якими обмінюються учасники системи обміну миттєвими повідомленнями. В серверному модулі ньому реалізована функція приєднання та (або) від'єднання клієнтів віл сситеми обміну миттєвими повідомленнями, надсилання текстових даних певному користувачеві та функція broadcast яка, як зрозуміло з назви, надсилає широкомовні повідомлення усім зареєстрованим користувачам системи.

Клас Server приведено на лістингу 2.1.

Лістинг 2.1 - Клас Server

```
import java.net.*;
import java.io.*;
import java.util.*;
public class Server implements Runnable {
private int port = 6564;
  private Hashtable idcon = new Hashtable();
  private int id = 0;
  static final String CRLF = "\r\n";
  synchronized void addConnection(Socket s) {
    ClientConnection con = new ClientConnection(this, s, id);
    id++;
  }
  synchronized void set(String the_id, ClientConnection con) {
    idcon.remove(the_id) ;
    con.setBusy(false);
    Enumeration e = idcon.keys();
    while (e.hasMoreElements()) {
      String id = (String)e.nextElement();
      ClientConnection other = (ClientConnection) idcon.get(id);
      if (!other.isBusy())
        con.write("add " + other + CRLF);
    }
```

```
    idcon.put(the_id, con);
    broadcast(the_id, "add " + con);
  }
  synchronized void sendto(String dest, String body) {
    ClientConnection con = (ClientConnection)idcon.get(dest);
    if (con != null) {
      con.write(body + CRLF);
    }
  }
  synchronized void broadcast(String exclude, String body) {
    Enumeration e = idcon.keys();
    while (e.hasMoreElements()) {
      String id = (String)e.nextElement();
      if (!exclude.equals(id)) {
        ClientConnection con = (ClientConnection) idcon.get(id);
        con.write(body + CRLF);
      }
    }
  }
  synchronized void delete(String the_id) {
    broadcast(the_id, "delete " + the_id);
  }
  synchronized void kill(ClientConnection c) {
    if (idcon.remove(c.getId()) == c) {
      delete(c.getId());
    }
  }
  public void run() {
    try {
      ServerSocket acceptSocket = new ServerSocket(port);
      System.out.println("Server listening on port " + port);
      while (true) {
        Socket s = acceptSocket.accept();
        addConnection(s);
      }
    } catch (IOException e) {
      System.out.println("accept loop IOException: " + e);
    }
  }
  public static void main(String args[]) {
    new Thread(new Server()).start();
    try {      Thread.currentThread().join();
    } catch (InterruptedException e) {
  }
  }
  }
```

Цей невеликий розроблений клас Server реалізовує серверну частину програми для системи обміну миттєвими повідомленнями. Точка входу в дану програму – це функція main. В даній програмі створюється головний потік, в якому створюється новий об’єкт стандартного класу ServerSocket. Цей новий об’єкт приєднується до певного порта (сокета) і в циклі програми здійснюється перевірка на підключення клієнта до цього порту чи сокету. У випадку наявності такого підключення створюється програмістом об’єкт типу ClientConnection, в якому реалізований потік по типовому опитуванню сокета, до якого підключений користувач системи обміну миттєвими повідомленнями, а також, тут відбувається обробка протокольних команд.

2.6.2 Розробка модуля, що реалізовує клас ServerConnection

В рамках розробки модуля запропонований клас ServerConnection є дзеркальним відображенням класу ClientConnection [37]. Але ServerConnection більш тісно переплетений саме з класом Client. В ServerConnection реалізований спеціальний потік, який опитує сокет, через який користувач системи обміну миттєвими повідомленнями під’єднаний до сервера, здійснює аналіз повідомлень, що надходять, а потім викликає необхідні методи класу Client. Крім того в класі ServerConnection є методи, які викликаються класом Client з метою відсилання даних на сервер лістинг 2.2.

Лістинг 2.2 - ServerConnection

```
import java.io.*;
import java.net.*;
import java.util.*;
class ServerConnection implements Runnable {
   void sendTo(String s, String id)
   {       if( id!= null )
      out.println("to "+id+" "+s);    }
  private static final int port = 6564;
  private static final String CRLF = "\r\n";
  private DataInputStream in;
```

```
  private PrintStream out;
  private String id, toid = null;
  private Client client;
  public ServerConnection(Scrabblet sc, String site) throws IOException {
  Socket server = new Socket(site, port);
  in = new DataInputStream(server.getInputStream());
  out = new PrintStream(server.getOutputStream(), true);   }
public ServerConnection( Client c,String site) throws IOException {
     client = c;
  Socket server = new Socket(site, port);
  in = new DataInputStream(server.getInputStream());
  out = new PrintStream(server.getOutputStream(), true);
  }
  private String readline() {
    try {
      return in.readLine();
    } catch (IOException e) {
      return null;
    }
  }
  void setName(String s) {
    out.println("name " + s);
  }
  void delete() {
    out.println("delete " + id);
  }
  void setTo(String to) {
    toid = to;
  }
  void send(String s) {
    if (toid != null)
      out.println("to " + toid + " " + s);
  }
  void chat(String s) {
    send("chat " + id + " " + s);
  }
void quit() {
    send("quit " + id);  // tell other player
    out.println("quit");
    stop();
  }
  private Thread t;
  void start() {
    t = new Thread(this);
    t.start();
  }
  void stop() {
    t.stop();
  }
  private static final int ID = 1;
```

```
  private static final int ADD = 2;
  private static final int DELETE = 3;
  private static final int CHAT = 4;
  private static final int QUIT = 5;
  private static Hashtable keys = new Hashtable();
  private static String keystrings[] = {
    "", "id", "add", "delete","chat",
    "quit"   };
  static {
    for (int i = 0; i < keystrings.length; i++)
      keys.put(keystrings[i], new Integer(i));
  }
  private int lookup(String s) {
    Integer i = (Integer) keys.get(s);
    return i == null ? -1 : i.intValue();
  }
  public void run() {
    String s;
    StringTokenizer st;
    while ((s = readline()) != null) {
      st = new StringTokenizer(s);
      String keyword = st.nextToken();
      switch (lookup(keyword)) {
      default:
        System.out.println("bogus keyword: " + keyword + "\r");
        break;
      case ID:
        id = st.nextToken();
        break;
      case ADD: {
          String id = st.nextToken();
          String hostname = st.nextToken();
          String name = st.nextToken(CRLF);
          client.add(id, hostname, name);
        }
        break;
      case CHAT: {
          String from = st.nextToken();
          client.chat(from, st.nextToken(CRLF));
        }
        break;
      case QUIT: {
          String from = st.nextToken();
          client.quit(from);
        }
        break;
      }
    }
  }
}
```

Даний модуль, модуля, що реалізовує клас ServerConnection легко може бути модифікованим в галузі розширення кількості протокольних команд та ускладнення ролі користувача системи обміну миттєвими повідомленнями за рахунок додавання специфічних реакцій на нові команди. В запропонованому класі ClientConnection може бути в подальшому ускладнена реакція на під'єднання користувача системи. Скажімо, користувачу можуть видаватися дані про початковий стан спільної задачі чи роботи. Таким чином наведена програма може служити скелетом для створення більш складних подібних систем. Можна дещо модифікувати класи Client та ServerConnection, залишивши в них лише основні функції з таокю метою, щоб потім розширювати саме їх шляхом механізму об'єктного успадкування.

2.6.3 Реалізація клієнтської частини

Реалізація клієнтської частини означена розробкою модуля, що реалізовує запропонований клас ClientConnection.

Запропонований автором в рамках даного завдання клас ClientConnection реалізує обмін даними з конкретним користувачем системи обміну миттєвими повідомленнями. В ClientConnection зберігається об'єкт класу Socket, до якого приєднаний цей користувач системи обміну миттєвими повідомленнями, далі створюється спеціальний потік, в якому здійснюється періодична спроба читати з потрібного нам сокета (лістинг 2.3).

Лістинг 2.3 – ClientConnection

```
import java.net.*;
import java.io.*;
import java.util.*;
class ClientConnection implements Runnable {
  private Socket sock;
  private DataInputStream in;
```

```
  private OutputStream out;
  private String host;
  private Server server;
  private static final int bufsize = 8192;
  private byte buffer[] = new byte[bufsize];
  private static final String CRLF = "\r\n";
  private String name = null;
  private String id;
  private boolean busy = false;
  public ClientConnection(Server srv, Socket s, int i) {
    try {
      server = srv;
      sock = s;
      in = new DataInputStream(s.getInputStream());
      out = s.getOutputStream();
      host = s.getInetAddress().getHostName();
      id = "" + i;                                    // tell the new one who it is...
      write("id " + id + CRLF);
      new Thread(this).start();
    } catch (IOException e) {
      System.out.println("failed ClientConnection " + e);    }
  }
public String toString() {
    return id + " " + host + " " + name;  }
  public String getHost() {
    return host;
  }
  public String getId() {
    return id;  }
  public boolean isBusy() {
    return busy;  }
  public void setBusy(boolean b) {
    busy = b;
  }
  public void close() {
    server.kill(this);
    try {
      sock.close();   // closes in and out too.
} catch (IOException e) {    }
  }
  public void write(String s) {
    byte buf[] = new byte[s.length()];
    s.getBytes(0, buf.length, buf, 0);
    try {
      out.write(buf, 0, buf.length);
    } catch (IOException e) {
      close();     }
  }
  private String readline() {
    try {       return in.readLine();
```

```
    } catch (IOException e) {
      return null;     }
  }
  static private final int NAME = 1;
  static private final int QUIT = 2;
  static private final int TO = 3;
  static private final int DELETE = 4;
  static private Hashtable keys = new Hashtable();
  static private String keystrings[] = {
    "", "name", "quit", "to", "delete"   };
  static {
    for (int i = 0; i < keystrings.length; i++)
      keys.put(keystrings[i], new Integer(i));
  }
  private int lookup(String s) {
    Integer i = (Integer) keys.get(s);
    return i == null ? -1 : i.intValue();
  }
  public void run() {
    String s;
    StringTokenizer st;
out:
    while ((s = readline()) != null) {
      st = new StringTokenizer(s);
      String keyword = st.nextToken();
      switch (lookup(keyword)) {
      default:
        System.out.println("bogus keyword: " + keyword + "\r");
        break;
      case NAME:
        name = st.nextToken() +
          (st.hasMoreTokens() ? " " + st.nextToken(CRLF) : "");
        System.out.println("[" + new Date() + "] " + this + "\r");
        server.set(id, this);
        break;
      case QUIT:
        break out;
      case TO:
        String dest = st.nextToken();
        String body = st.nextToken(CRLF);
        server.sendto(dest, body);
        break;
      case DELETE:
        busy = true;
        server.delete(id);
        break;       }
    }
    close();   }
}
```

Якщо ця спроба підключитися завершується успіхом, тобто надійшло повідомлення від користувача системи обміну миттєвими повідомленнями, це повідомлення надалі аналізується і здійснюється відповідна реакція на нього.

З метою отримання цієї реакції викликаються методи класу Server, об’єкт якого передається новоствореному класу ClientConnection як параметр конструктора. Крім того в класі ClientConnection є метод, який здійснює безпосереднє надсилання даних до користувача системи обміну миттєвими повідомленнями. Коли серверу потрібно надіслати дані конкретному користувачу системи обміну миттєвими повідомленнями, він викликає цей метод.

2.6.4 Розробка модуля, що реалізовує клас Client

Даний клас Client є розширенням класу Applet, отже є аплетом також. В класі Client створюється об’єкт класу ServerConnection, через якій здійснюється обмін даними з сервером. Графічний інтерфейс класу ServerConnection представляє такі поля:

1) поле вводу, в яке спочатку вводиться ім’я користувача,
2) повідомлення, адресовані іншим клієнтам,
3) список клієнтів, в якому треба обирати адресата
4) текстова область в якій відбивається текст діалога (лістинг 2.4).

Лістинг 2.4 – Клас Client

```
import java.util.*;
import java.io.*;
import java.net.*;
import java.awt.*;
import java.applet.*;
public class Client extends Applet {
  private ServerConnection server;
  private String serverName;
  private boolean single = false;
  private boolean seen_pass = false;
  private boolean name_set = false;
  private String name;
  private String others_name;
  private Panel topPanel;
  private Label prompt;
```

```
private TextField namefield;
private Button done;
private TextField chatfield;
private List idList;
private TextArea dialogArea;
public void init() {
  setLayout( new BorderLayout() );
  serverName = getCodeBase().getHost();
  if (serverName.equals(""))
    serverName = "localhost";
  prompt = new Label("Enter id:");
  namefield = new TextField(30);
  topPanel = new Panel();
  topPanel.setBackground(new Color(255, 255, 200));
  topPanel.add(prompt);
  topPanel.add(namefield);
  add("North", topPanel);
  idList = new List(10, false);
  add("West", idList );
  dialogArea = new TextArea();
  dialogArea.setEditable( false );
  add("Center", dialogArea );
}
public void start() {
  try {
    showStatus("Connecting to " + serverName);
    server = new ServerConnection(this,serverName);
    server.start();
    showStatus("Connected: " + serverName);
  } catch (Exception e) {
    single = true;    }
}
public void stop() {
  if (!single)
    server.quit();  }
void add(String id, String hostname, String name) {
  delete(id); // in case it is already there.
  idList.addItem("(" + id + ")  " + name + "@" + hostname);
}
void delete(String id) {
  for (int i = 0; i < idList.countItems(); i++) {
    String s = idList.getItem(i);
    s = s.substring(s.indexOf("(") + 1, s.indexOf(")"));
    if (s.equals(id)) {
      idList.delItem(i);
      break;       }
  }
  if (idList.countItems() == 0)
    showStatus("Wait for other players to arrive.");
}
```

```
private String getName(String id) {
  for (int i = 0; i < idList.countItems(); i++) {
    String s = idList.getItem(i);
    String id1 = s.substring(s.indexOf("(") + 1, s.indexOf(")"));
    if (id1.equals(id)) {
      return s.substring(s.indexOf(" ") + 3, s.indexOf("@"));
    }
  }
  return null;
}
void chat(String id, String s) {
  dialogArea.appendText(id + ": " + s+ "\n");//append
  showStatus(id + ": " + s);
}
void quit(String id) {
  showStatus(id + " just quit.");
  delete(id);
}
private void nameEntered(String s) {
  if (s.equals(""))
    return;
  System.out.println(s);
  name = s;
  if( !single )
  {
     server.setName(name);
     showStatus("Wait for other players to arrive.");
  }
  prompt.setText("You say:");
  name_set = true;
}
public boolean action(Event evt, Object arg) {
  System.out.println("a");
 if(evt.id == Event.ACTION_EVENT)
 if (evt.target == namefield){
    if( name_set ){
      dialogArea.appendText(name + ": " + namefield.getText()+"\n");
      if(!single)
      server.sendTo((String)arg, idList.getSelectedItem());
    }
    else nameEntered((String)arg);
    namefield.setText("");
 }
  return true;
 }
}
```

Розроблений метод action реалізує обробку події введення текстових даних в область введення даних.

Розроблені методи add, delete, chat, nameEntered, quit є реакцією на такі події:

1) підключення нового клієнта;
2) зникнення клієнта;
3) надходження повідомлення від клієнта;
4) ідентифікація;
5) вихід.

Ці методи викликаються програмістом і запропонованим класом ServerConnection під час надходження відповідних повідомлень чи запитів, чи викликаються всередині самого аплета, якщо джерелом події є сам користувач системи обміну миттєвими повідомленнями.

2.6.5 Розробка додаткового функціоналу

Важливиою частиною програми, написаної на Java є підключені до виконуваного файлу модулів та підпрограм з використанням принципів наслідування та поліморфізму. Нижче приведено фрагмент лістингу із зазначенням підключених фрагментів [38-40] (лістинг 2.5).

Лістинг 2.5 – Підключені модулі

```
package ua.vinnica.vntu.taskmsg;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Vector;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
```

```
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;
import javax.swing.table.DefaultTableModel;
import ua.vinnica.vntu.taskmsg.ui.MessageDialog;
import ua.vinnica.vntu.tools.Utilities;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import ua.vinnica.vntu.taskmsg.ui.SelectModeDialog;
import ua.vinnica.vntu.taskmsg.ui.TablePane;
```

З метою, аби визначити, який варіант програми для системи обміну миттєвими повідомленнями повинен бути запущений, використаємо такий код (лістинг 2.6):

Рисунок 2.6 – Варіант програми для вибору

```
SelectModeDialog dialog = new SelectModeDialog(this);
 dialog.show();
 if(!dialog.accepted)
  System.exit(0);
```

В залежності від того, який режим вибрано [41], можемо тепер сформувати користувацький інтерфейс системи обміну миттєвими повідомленнями (лістинг 2.7):

Лістинг 2.7 – Формування користувацького інтерфейсу

```
if(dialog.checkServerMode())
{    makeServerUI();
 mode = true;
}
else
 makeClientUI();
listener.start();
addWindowListener(new WindowAdapter()
{    public void windowClosing(WindowEvent e)
 {
  System.exit(0);     }
});
```

```
}
  public static void main(String args[])
{   TaskMessenger messenger = new TaskMessenger();
 messenger.show();
}
```

Код для редагування списку спільних та задач чи робіт приведено на лістингу 2.8.

Лістинг 2.8 – Редагування списку повідомлень

```
public void mouseClicked(MouseEvent e) {
if ((e.getButton() == MouseEvent.BUTTON3) && (e.getClickCount() == 1)) {
JPopupMenu contextMenu = new JPopupMenu("Рядки");
contextMenu.add(Utilities.createMenuItem("Новий рядок", "",
"addRow", source));
contextMenu.add(Utilities.createMenuItem("Додати кілька рядків",
"", "addMany", source));
contextMenu.add(Utilities.createMenuItem("Видалити рядок", "",
"removeRow", source));
contextMenu.show((JComponent) e.getSource(), e.getX(), e.getY());   }
}
}
```

2.6.6 Реалізація користувацького інтерфейсу

Користувацький інтерфейс програми наведено нижче (рисунки 2.10 -2.15).

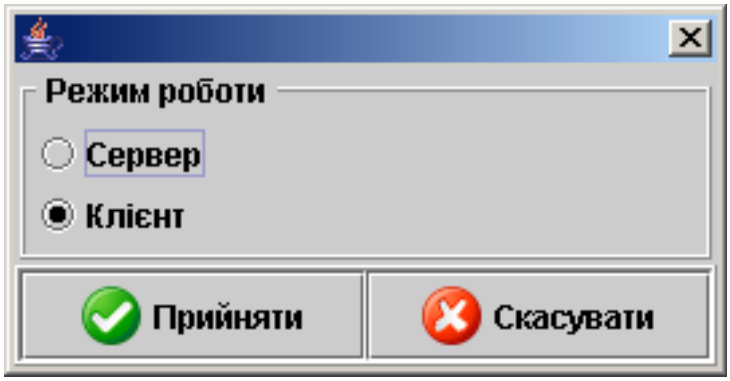

Рисунок 2.10 – Вікно вибору режиму програми

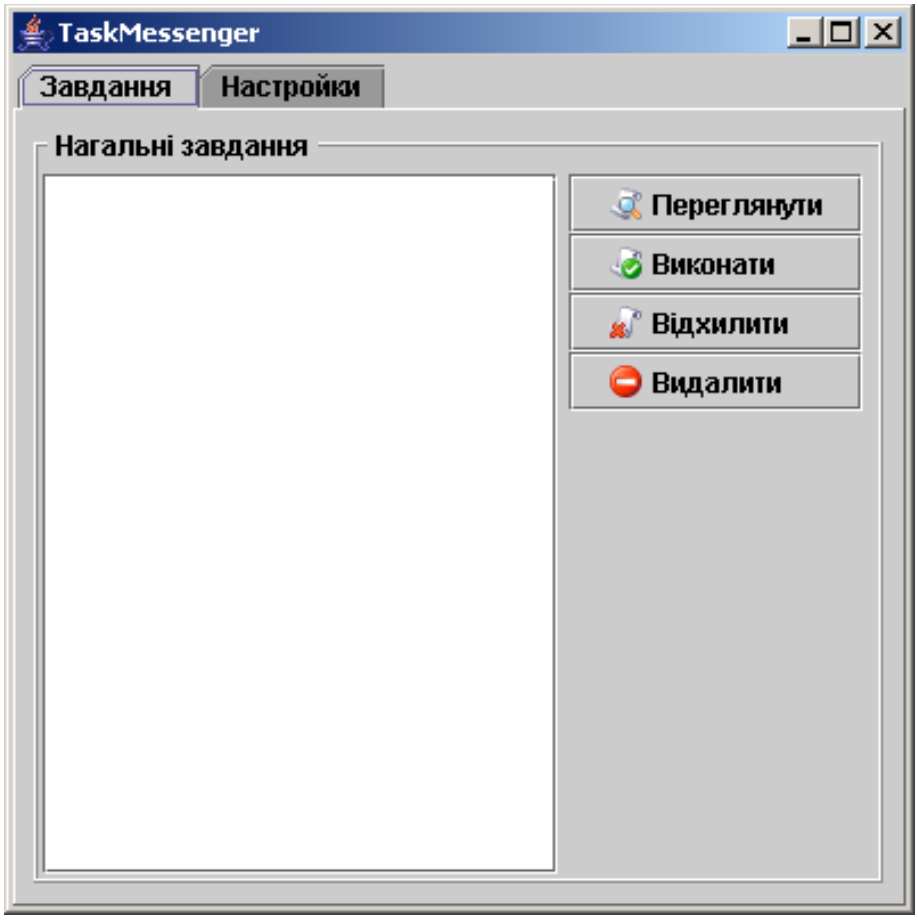

Рисунок 2.11 – Основне вікно серверної частини

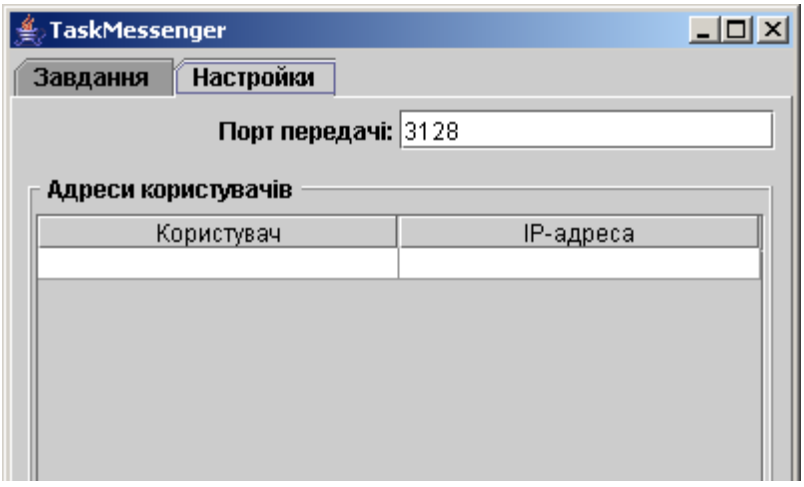

Рисунок 2.12 – Вікно налагодження серверної частини

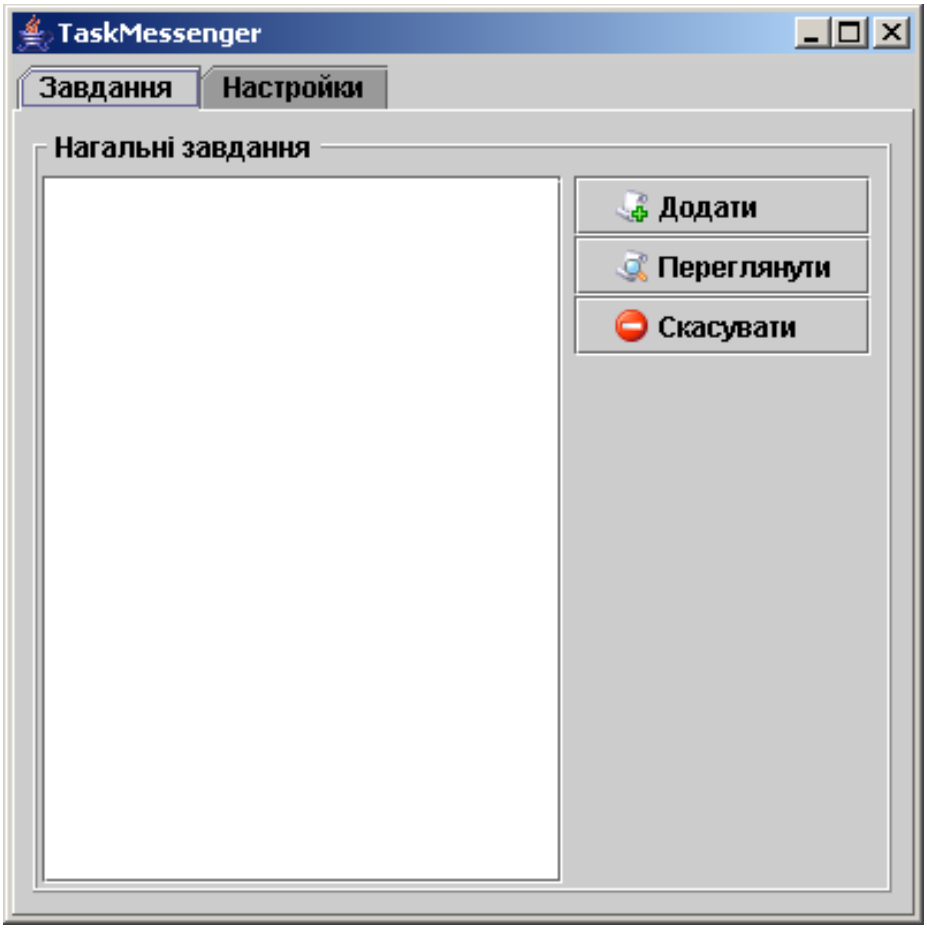

Рисунок 2.13 – Вікно клієнтської частини

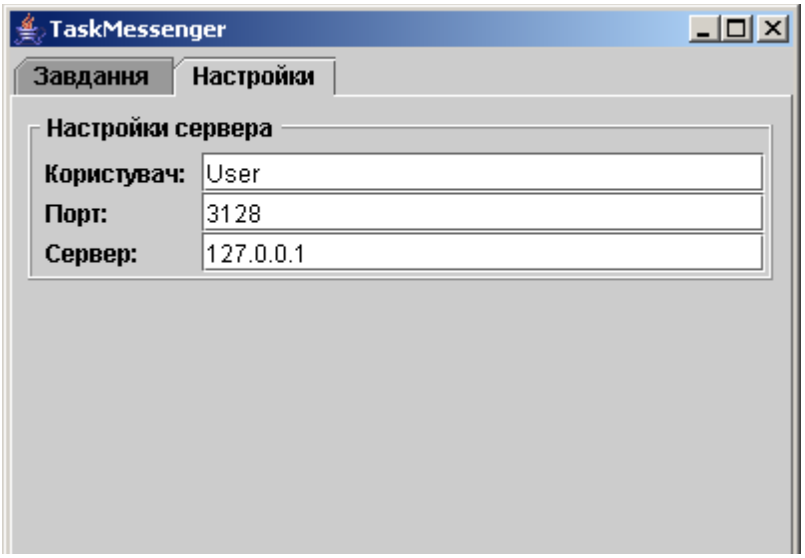

Рисунок 2.14 – Вікно налагодження клієнтської частини

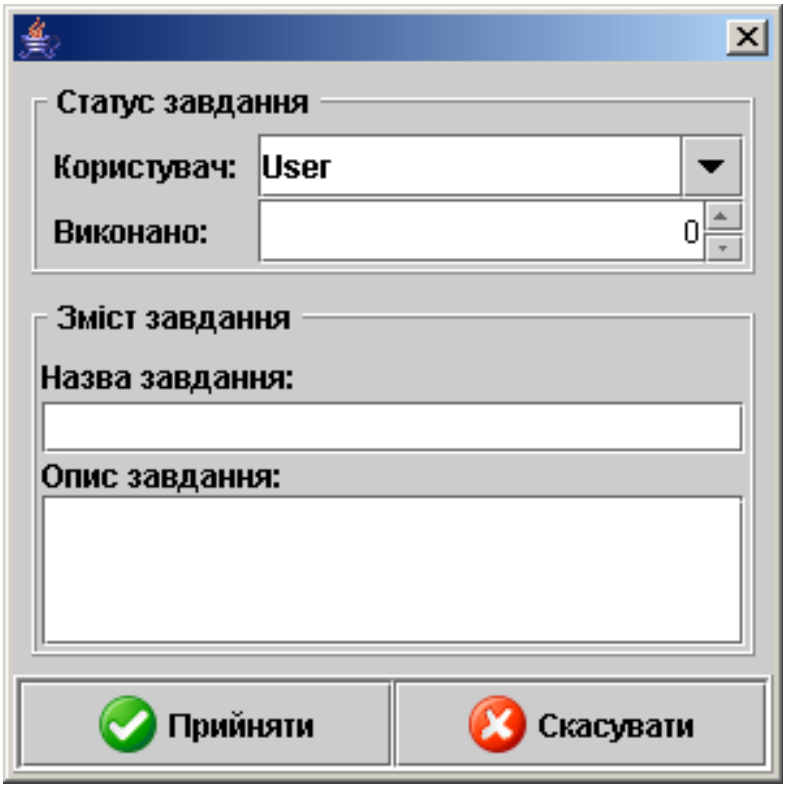

Рисунок 2.15 – Вікно властивостей спільних задач

2.7 Тестування реалізація системи обміну миттєвими повідомленнями

Для проведення тестування програмного засобу для системи обміну миттєвими повідомленнями необхідно:

1) Встановити середовище jre (каталог jre), причому, можна протестувати дану системи обміну миттєвими повідомленнями на одному комп’ютері.

2) Далі необхідно запустити спеціальний файл TaskMessenger.bat.

3) Вибрати спосіб, у якому режимі запускатиметься програма для системи обміну миттєвими повідомленнями (рис. 2.15).

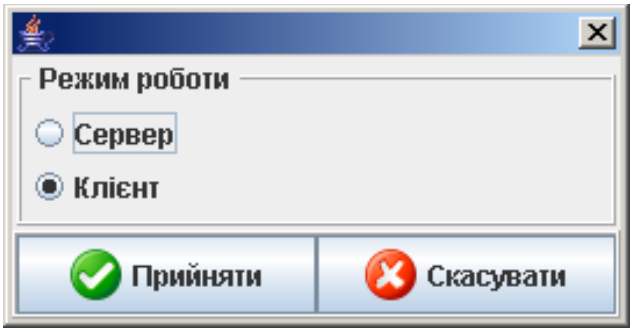

Рисунок 2.16 – Вибір режиму запуску програми «Клієнт»

Якщо вибраний клієнтський режим у середовищі системи обміну миттєвими повідомленнями, то він може ставити задачі серверу за допомогою кнопки «Додати» (рис. 2.17).

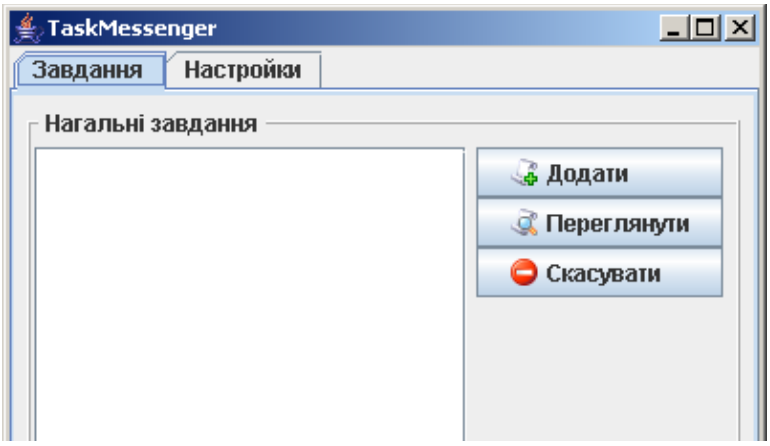

Рисунок 2.17 – Тестування кнопки «Додати»

При цьому використовується діалогове вікно властивостей системи обміну миттєвими повідомленнями, яке показано на рис. 2.17 для завдання початкових параметрів задачі.

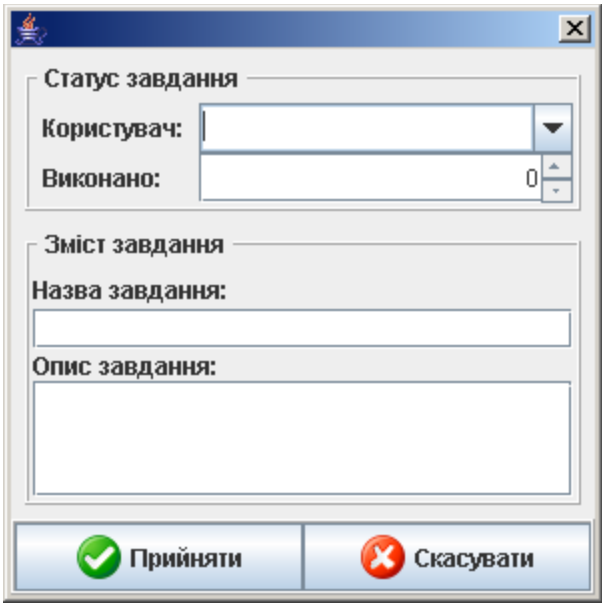

Рисунок 2.18 – Тестування діалогового вікна властивостей

Якщо в системі обміну миттєвими повідомленнями обрано «Сервер» (рис. 2.5), то він може рапортувати про виконання задачі за допомогою кнопок (рис. 2.19)

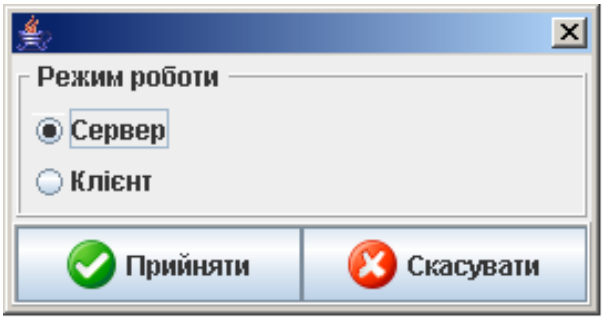

Рисунок 2.19 – Вибір режиму запуску програми «Сервер»

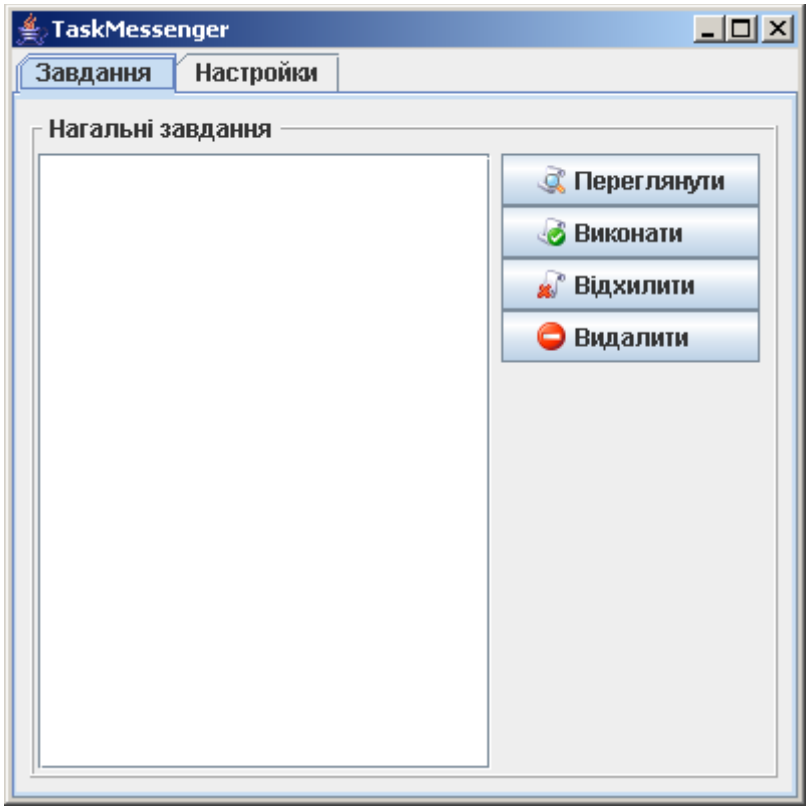

Рисунок 2.20 – Тестування рапорту про виконання задачі

Зміни у цьому діалоговому вікні системи обміну миттєвими повідомленнями (рис. 2.20) будуть якнайшвидше відображені на стороні клієнта. Крім того, за допомогою кнопок «Виконати» можна автоматично присвоїти спільній задачі 100% готовність, а за допомогою кнопки «Відхилити» - відповідно, відхилити задачу. Це також буде миттєво відображено на стороні клієнта системи обміну миттєвими повідомленнями.

Налаштування. В режимі сервера системи обміну миттєвими повідомленнями нічого налаштовувати не потрібно. А в режимі налаштування клієнта налаштовується IP-адреса сервера і робочий порт системи обміну миттєвими повідомленнями, через який і відбувається обмін миттєвими повідомленнями (рис. 2.21).

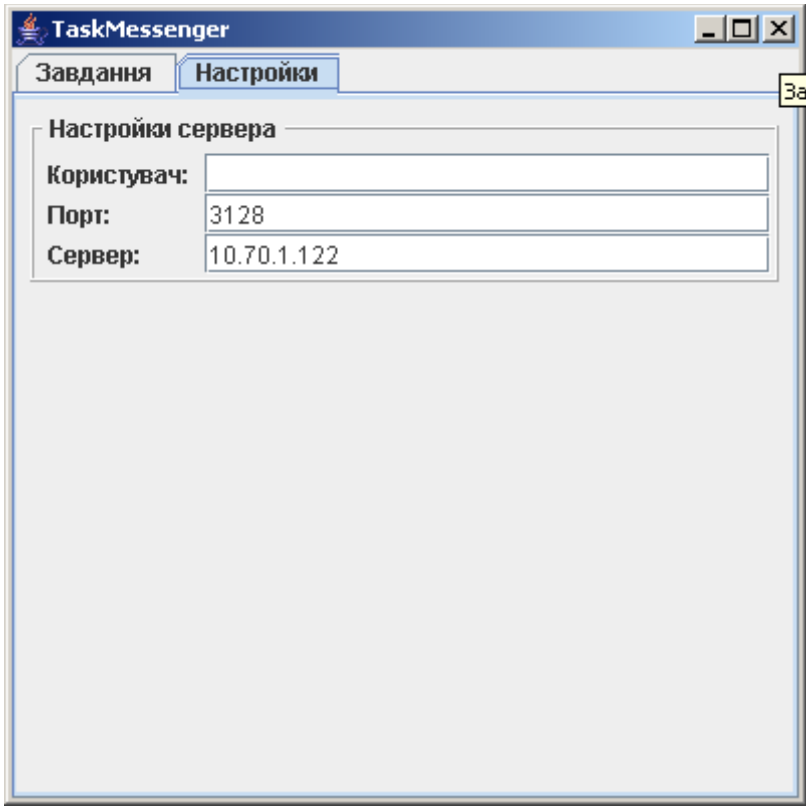

Рисунок 2.21 – Налаштування доступу до сервера

2.8 Висновки за розділом

В даному розділі створено схему роботи системи обміну миттєвими повідомленнями, робота якої полягає у обміні даними між двома основними частинами цієї системи: підсистеми обміну миттєвими повідомленнями і підсистеми контролю виконання спільних задач (робіт). Розроблено власний тип даних за допомогою технології XML для процесу обміну повідомленнями в межах цієї системи. Запропоновано модель обміну миттєвими повідомленнями в корпоративній мережі, алгоритм обміну миттєвими повідомленнями в корпоративній мережі на стороні клієнта та сервера і алгоритм командного контролю виконання задач. Виконана програмна реалізація та тестування роботи системи обміну миттєвими повідомленнями.

3 ЕКОНОМІЧНА ЧАСТИНА

3.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності [42].

Результатом магістерської кваліфікаційної роботи «Система обміну миттєвими повідомленнями в корпоративній мережі» є розробка програми для обміну повідомленнями в корпоративній мережі. Для проведення технологічного аудиту залучено трьох незалежних експертів. У нашому випадку такими експертами є: Колесник Ірина Сергіївна (к.т.н., доцент каф. обчислювальної техніки ВНТУ), Черняк Олександр Іванович (к.т.н., доцент каф. обчислювальної техніки ВНТУ), Снігур Анатолій Васильович(к.т.н., доцент каф. обчислювальної техніки ВНТУ).

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 3.1.

Таблиця 3.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища	Ціна продукту дещо вища за	Ціна продукту приблизно	Ціна продукту	Ціна продукту значно нижче за

	за ціни аналогів	ціни аналогів	дорівнює цінам аналогів	дещо нижче за ціни аналогів	ціни аналогів

Продовження таблиці 3.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер.	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 3.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 3.2.

Таблиця 3.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Колесник	2 – Черняк	3 – Снігур
	Бали, виставлені експертами:		
1	3	3	4
Ринкові переваги (недоліки):			
2	3	2	4
3	4	4	4
4	4	3	4
5	3	2	3
Ринкові перспективи			
6	2	2	2

7	4	4	3
Практична здійсненність			
8	3	3	2
9	4	4	3
10	4	4	4
11	3	4	3
12	3	3	3
Сума балів	$СБ_1$=40	$СБ_2$=38	$СБ_3$=39
Середньоарифметична сума балів $\overline{СБ}$	39		

За даними таблиці 3.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 3.3.

Таблиця 3.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 39 балів, що відповідає рівню «вище середнього».

Одним з сучасних видів є спеціалізовані засоби, методики та підходи до процесів корпоративного спілкування за допомогою програмних засобів та корпоративних комп’ютерних мереж. Це говорить про доцільність та необхідність створення такої системи обміну миттєвими повідомленнями в корпоративній мережі, яка би поєднувала у собі простоту інтерфейсу універсального месенджера і функції спеціалізованого месенджера, а також, такої, яка би дозволила користувачам ефективно вести обмін миттєвими повідомленнями в корпоративній мережі та одночасно мати функціонал контролю за виконанням спільних задач чи робіт.

Всі існуючі нині програми обміну миттєвими повідомленнями можна поділити на дві великі категорії [10-13]:

- універсальні;
- спеціалізовані.

Універсальні носять уніфіковану назву месенджерів, і призначені для обміну довільними миттєвими повідомленнями у текстовому вигляді. У свою чергу, спеціалізовані ж побудовані на основі певного, наперед визначеного спеціалізованого формату даних, який формує «конверт» (т.з., envelope) і містить в собі лише необхідні користувачеві дані. Нині на ринку існуюпредставленість такі універсальні месенджери як:

- ICQ;
- AOL Instant Messenger;
- MSN Messenger;
- Jabber;
- Apple Bonjour.

Всі перелічені вище універсальні месенджери мають майже один і той самий графічний інтерфейс, що має перелік користувачів на основній панелі програмного засобу, і набір ряду функцій, що забезпечують власне обмін миттєвими повідомленнями.

Відомо, що чіткого поділу між серверною та клієнтською складовими в цих програмних засобах немає, оскільки вони одночасно виконують роль і сервера (тобто, приймача повідомлень) і клієнта (тобто, передавача повідомлень).

В якості аналога для розробки було обрано месенджер Jabber. Основними недоліками аналога є:

- незручна система передавання файлів (через контакт-лист);
- проблеми з процесами кодуванням;

– аватар має бути не більшим за 64 на 64 пікселів.

У таблиці 3.4 наведені основні технічні показники аналога і нового програмного продукту

Робота була присвячена підвищенню рівня контролю виконання спільних задач під час процесів передачі миттєвих повідомлень в корпоративних мережах за допомогою створення системи обміну миттєвими повідомленнями в корпоративній мережі, яка додатково дозволяє дає можливість кваліфікованого опису стану вирішення задачі чи роботи в системах колективного виконання задач. Мета розробки - створення додаткового засобу зв’язку, який б допомагав формалізувати подібні випадки за допомогою механізму обміну миттєвими повідомленнями в корпоративній мережі до такого рівня, аби професіонал-інженер міг ефективно визначати проблему на основі словесної інформації від користувача.

Таблиця 3.4 - Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Функціональність	3	5	5/3
Надійність	4	4	4/4
Сумісність	3	3	3/3
Супровід	3	4	3/4
Простота використання	4	5	4/5

Запропонована система обміну миттєвими повідомленнями в корпоративній мережі поєднує у собі простоту інтерфейсу універсального месенджера і функції спеціалізованого месенджера, а також, дозволяє користувачам ефективно вести обмін миттєвими повідомленнями в

корпоративній мережі та одночасно мати функціонал контролю за виконанням спільних задач чи робіт.

Продукт, що пропонується є якісно новим, так як пропонує кінцевому споживачу набір функцій, які на даний момент повністю не пропонуються кінцевому споживачу у жодному з існуючих на ринку аналогів.

Розповсюдження програмного забезпечення є умовно-безкоштовним (Trial), тобто користувач може протестувати програму на протязі визначеного періоду часу, при чому для безкоштовного користування надається обмежена версія програмного забезпечення.

Для просування програмного додатку на ринку планується розробка власного сайту з його подальшим просуванням у рейтингу видачі найбільших пошукових систем. Крім того планується закупка цільової реклами на найбільших сайтах з розміщення та обробки цифрових зображеннях.

3.2 Прогнозування витрат на виконання наукової роботи та впровадження результатів.

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення, яке складається з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

2-й етап: розрахунок загальних витрат на виконання даної роботи;

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат приймаючи до уваги те, що розробкою займався один розробник програмного забезпечення.

1. Основна заробітна розробника-дослідника $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t \text{ [грн]}, \quad (3.1)$$

де М – місячний посадовий оклад – 8500 грн;

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (22) дні;

t – число робочих днів роботи розробника-дослідника - 66.

$$З = 8500/22 \cdot 66 = 25500{,}00 \text{ (грн)}.$$

2. Додаткова заробітна плата $З_д$ розробника розраховується як 10% від основної заробітної плати:

$$З_д = 0{,}10 \cdot 25500{,}00 = 2550{,}00 \text{ (грн)}.$$

3. Нарахування на заробітну плату $Н_{зп}$ розробника становить:

$$H_{зп} = (З_о + З_о) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (3.2)$$

де $З_о$ – основна заробітна плата розробника;

$З_д$ – додаткова заробітна плата розробника;

β – ставка єдиного соціального внеску – 22%.

$$Н_{зп} = (25500,00 + 2550,00) \cdot 0,22 = 6171,00 \text{ (грн)}.$$

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи. Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому розраховуємо за формулою:

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В} \text{ [грн]}, \qquad (3.3)$$

де Ц – загальна балансова вартість обладнання, приміщення тощо, грн;

Т – фактична тривалість використання, міс;

$Т_В$ – термін використання обладнання, приміщень тощо, роки.

Розробка програмного забезпечення проводилася протягом 3 місяців.

Зроблені розрахунки зведено до таблиці 3.5.

Таблиця 3.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, р	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	120000	25	3	100
Комп'ютер	15000	5	3	750
Монітор	6700	6	3	279
Всього				*1129*

Інформацію про матеріали, що використовуються при розробці продукту внесено до таблиці 3.6.

Таблиця 3.6 – Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу, грн
Папір (пачка)	78,00	1	78,00
Ручка	6,00	2	12,00
Всього			90,00

Витрати на енергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули 3.2:

$$B_e = B \cdot П \cdot Ф \cdot K_п \ [грн], \tag{3.4}$$

де В – вартість 1кВт електроенергії;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи комп'ютера при створенні програмного продукту, годин;

$К_п$ – коефіцієнт використання потужності.

$$B_e = 1{,}88 \cdot 0{,}5 \cdot 528 \cdot 0{,}4 = 198{,}53 \ (грн).$$

Витрати на доступ до Інтернет можна розрухувати за формулою:

$$B_{дi} = Ц_{дi} \cdot T \ [грн], \tag{3.5}$$

де $Ц_{ді}$ – ціна доступу за місяць;

Т – кількість місяців використання доступу до мережі.

Отже, витрати на доступ до мережі Інтернет становлять:

$$В_{ді} = 150 \cdot 3 = 450 \text{ (грн)}.$$

Інші витрати становлять 50% від основної заробітної плати розробника, і дорівнюють – 12750 грн.

Сума усіх витрат, що вказані вище дає витрати на виконання даного етапу роботи В:

$$В = З^{о}+З^{д}+Н^{зп}+А + В^{мат} + В^{е} + В^{ді} \text{ [грн]},$$

$$В = 25500+2550+6171 + 1129,00 + 90,00 +198,53+450,00+12750=48838,53$$

(грн).

2-й етап. Розрахунок загальних витрат на виконання даної роботи. Загальна вартість всієї наукової роботи визначається за $В_{заг}$ формулою:

$$B_{заг} = \frac{B}{\alpha} \text{ [грн]}, \quad (3.6)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях.

Так, як над роботою задіяна одна людина, якою виконується уся робота, то α становить 1. Підставивши дані у формулу, отримуємо:

$$В_{заг} = 48838,53 \text{ (грн)}.$$

3-й етап. Прогнозування загальних витрат на виконання та впровадження результатів виконаної роботи.

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \ [грн], \tag{3.7}$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Розробка знаходиться на стадії розробки дослідного зразка, то $\beta \approx 0,5$;

Отже, підставимо дані в формулу й отримаємо результат:

$$ЗВ = \frac{48838,53}{0,7} = 69769,32 \text{ (грн)}.$$

3.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спробуємо кількісно спрогнозувати, яку вигоду, можна отримати у майбутньому від впровадження результатів виконаної наукової роботи.

Всі зроблені розрахунки є приблизними і не передбачають деталізації.

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства (організації). Зростання чистого прибутку ми можемо оцінити у теперішній вартості грошей.

Саме зростання чистого прибутку забезпечить підприємству надходження додаткових коштів, які дозволять покращити фінансові результати діяльності та виплатити кредити (якщо вони потрібні для впровадження результатів розробки).

Оцінити збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки можливо використовуючи формулу:

$$\Delta П_i = \sum_i^n \left(\Delta П_o \cdot N + Ц_o \cdot \Delta N\right) \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) [грн], \quad (3.8)$$

де ΔПо – покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ – коефіцієнт, який враховує сплату податку на додану вартість.

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = 0{,}2...0{,}3$;

υ – ставка податку на прибуток (18%).

В результаті впровадження результатів наукової розробки покращується якість продукту, що дозволяє підвищити ціну його реалізації на 500 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 1000 шт., протягом другого року – ще на 500 шт., протягом третього року – ще на 1000 шт.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 1 шт., а її ціна – 2000 грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства ΔПі протягом першого року складе:

$$\Delta П_1 = [2000 \cdot 1 + 2500 \cdot 100] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 250200\ (грн).$$

Збільшення чистого прибутку підприємства ΔПі протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_2 = \left[[2000 \cdot 1 + 2500 \cdot 150]\right] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 375200\ (грн).$$

Збільшення чистого прибутку підприємства ΔПі протягом третього року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_3 = [[2000 \cdot 1 + 2500 \cdot 200]] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 500200\ (грн).$$

3.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1-й крок. Розраховується теперішня вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану за формулою, тобто будемо вважати, що ЗВ = PV=69769,32

2-й крок. Розраховується очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане нами раніше.

3-й крок. Будуємо вісь часу, на яку наносимо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі показуємо у ті терміни, коли вони здійснюються.

Припустимо, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнює 69769,32 грн. Результати вкладених у наукову розробку інвестицій повинні почати окупатись протягом трьох років. У першому році підприємство отримає збільшення чистого прибутку в сумі 250200,00 грн відносно базового року, у другому році – збільшення чистого прибутку на 375200,00 грн (відносно базового року), у третьому році – збільшення чистого прибутку 500200,00 грн (відносно базового року).

Тоді рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 3.1.

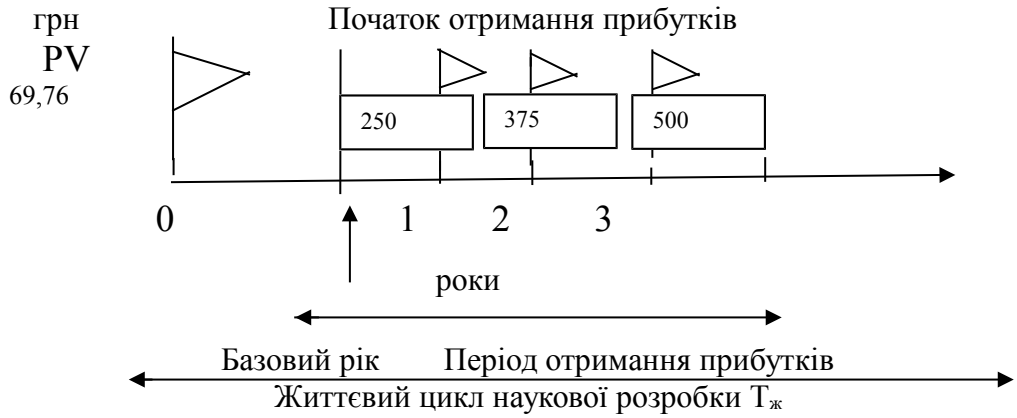

Рисунок 3.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4-й крок. Розраховується абсолютна ефективність вкладених інвестицій $Е_{абс}$.

Для цього використовується формула:

$$Е_{абс} = (ПП - PV), \quad (3.9)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (3.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

Якщо $Е_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Ставка дисконтування τ дорівнює 0,1. Отримаємо:

$$ПП = \frac{250200}{(1+0{,}1)^1} + \frac{375200}{(1+0{,}1)^2} + \frac{500200}{(1+0{,}1)^3} = 879180{,}52\ (грн).$$

Тоді,

$$Е_{абс} = 879180{,}52 - 69769{,}32 = 809411{,}20\ (грн).$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР є доцільним.

5-й крок. Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього використовуємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (3.11)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV –теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Далі, розрахована величина $E_в$ порівнюється з мінімальною (бар'єрною) ставкою дисконтування τ мін, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ мін визначається за формулою 3.10:

$$\tau = d + f, \quad (3.12)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,20;

f – показник, що характеризує ризикованість вкладень, f = 0,05.

Якщо величина $E_в > \tau$ мін, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде.

Спрогнозуємо величину τ мін. Припустимо, що за даних умов τ мін = 0,2 + 0,05 = 0,25. Тоді відносна (щорічна) ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$E_в = \sqrt[3]{1 + \frac{809411,20}{69769,32}} - 1 = \sqrt[3]{12,6} - 1 = 1,32 \text{ або } 132\%$$

Оскільки $E_в$ = 132% > τ мін = 0,25 = 25%, у інвестора буде зацікавленість вкладати гроші в дану наукову розробку, оскільки значно більші прибутки він отримає від того, що інвестує кошти розробку, а не розмістить гроші на депозиті у комерційному банку.

6-й крок. Розраховуємо термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою:

$$T_{ок} = \frac{1}{E_{в.}} \quad (3.13)$$

Для нової розробки термін окупності вкладених у реалізацію проекту інвестицій $T_{ок}$ складе:

$$T_{ок} = \frac{1}{1,32} = 0,75\ (року),$$

Оскільки $T_{ок}$ менше 3-х років, то фінансування даної наукової розробки є доцільним.

3.5 Висновки за розділом

В даному розділі було виконано оцінювання комерційного потенціалу розробки програми для обміну миттєвими повідомленнями в корпоративній мережі.

Проведено технологічний аудит з залученням трьох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки високий.

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти і є перспективною розробкою. Він має кращі функціональні

показники, а тому є конкурентоспроможним товаром на ринку. Існуючі переваги нової розробки дозволять швидко її поширити на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 69769,32 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 809411,20 грн свідчить про отримання прибутку інвестором при комерціалізації програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 132%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 70%. Це означає можливість потенційної зацікавленості інвесторів у фінансуванні розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить 0,75 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

Дана магістерська кваліфікаційна робота була присвячена підвищенню рівня контролю виконання спільних задач під час процесів передачі миттєвих повідомлень в корпоративних мережах за допомогою створення системи обміну миттєвими повідомленнями в корпоративній мережі, яка додатково дозволяє дає можливість кваліфікованого опису стану вирішення задачі чи роботи в системах колективного виконання задач.

В цілому, наукова новизна одержаних в магістерській кваліфікаційній роботі результатів полягає в такому:

1) вдосконалено систему обміну миттєвими повідомленнями в корпоративній мережі, що використовує вдосконалений тип даних;
2) вдосконалено модель для процесу обміну повідомленнями, що дозволяє вести обмін миттєвими повідомленнями в корпоративній мережі та одночасно мати функціонал контролю за виконанням спільних задач чи робіт.

Запропонована система обміну миттєвими повідомленнями в корпоративній мережі поєднує у собі простоту інтерфейсу універсального месенджера і функції спеціалізованого месенджера, а також, дозволяє користувачам ефективно вести обмін миттєвими повідомленнями в

корпоративній мережі та одночасно мати функціонал контролю за виконанням спільних задач чи робіт.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Amble S.r Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process [Text] / Scott Ambler. – John Wiley & Sons, Inc., New York, 2002. – 402 p. – ISBN: 0-471-20282-7.
2) Herbsleb James D. Geographically Distributed Software Development. Bell Labs, Lucent Technologies [Electronic resource] / James D. Herbsleb Rebecca E. Grinter, and Lawrence Votta Jr. – Mode of access: http://kluedo.ub.uni-kl. de/volltexte/2000/217/pdf/no_series_210.pdf. – Title from the screen
3) Markovets O. Modeling of citizen claims processing by means of queuing system / O. Markovets, A. Peleschyshyn // International Journal of Computer Science and Business Informatics (IJCSBI). – Vol. 15, No. 1. – India : IJCSBI.ORG, 2015. – P. 36–46.
4) Sutherland Jeff. Distributed Scrum: Agile Project Management with Outsourced Development Teams [Electronic resource] / Sutherland Jeff, Ph. D. Anton Viktorov. – Mode of access: http://jeffsutherland.com/SutherlandDistributedScrumHIC CS2007.pdf. – Title from the screen.
5) Fedushko S. Design of registration and validation algorithm of member's personal data / S. Fedushko, Yu. Syerov // International Journal of Informatics and Communication Technology. – Indonesia: Institute of Advanced Engineering and Science, 2013. – Vol. 2. – No. 2. – P. 93–98.
6) Аминов И. И. Психология делового общения. – М.: Издательство: Омега-Л, 2006. - 304 с.
7) Винер Н. Кибернетика или управление и связь в животном и машине. – М.: Наука, 1983. – 340 с.
8) Дмитренко М.Й. Ділове спілкування як феномен соціальної дійсності: Автореф. дис... канд. філософ. наук: 09.00.03 / Харк. ун-т повітр. сил. — Х., 2005. — 19 с.

9) Колесникова Н.Л. Деловое общение. Business Communication: Учебное пособие. - М.: Флинта: Наука, 2007 – 152 с.
10) «Коммунікація» (визначення поняття)// http://ru.wikipedia.org/wiki
11) Корпоративная культура делового общения: Главные правила общения и поведения в современном обществе/ Авт-сост. И. Н. Кузнецов. – М.: АСТ; Мн.: Харвест. 2005. С. 442 – 533.
12) Лэйхифф Дж.М., Пенроуз Дж.М. Бизнес-коммуникации: Стратегии и навыки. - СПб.: Питер, 2001. - 686 с.
13) Мирошниченко А. Бизнес-коммуникации. Мастерство делового общения. – М.: Издательство «Книжный мир», 2008. – 384 с.
14) Назаренко В.О. Психологічні умови розвитку комунікативної компетентності керівників Державної прикордонної служби України: Автореф. дис... канд. психол. наук: 19.00.09 / Нац. акад. Держ. прикордон. служби України ім. Б.Хмельницького. — Хмельницький, 2007. — 19 с.
15) Орлова Т.М Коммуникационный менеджмент в управлении экономическими системами: Автореф. дис. … доктора экон. наук.: 080604 / Московский государственный университет. - М., 2003 – 43 с.
16) Панфилова А.П. Деловая коммуникация в профессиональной деятельности: Учебное пособие.-2-е изд. - СПб.: Знание, ИВЭСЭП, 2004. - 495 с.
17) Почепцов Г.Г. Теория коммуникации. - М.: Смартбук, 2008. – 656 с.
18) Спивак В.А. Современные бизнес-коммуникации. — СПб.: Питер, 2001. — 448 с.
19) Тер-Минасова С.Г. Язык и межкультурная коммуникация. - М.: Слово/Slovo, 2008. - 624 с.
20) Business Communication by A.C. "Buddy" Krizan, Patricia Merrier, Joyce P. Logan, and Karen Schneiter Williams - Publisher: SouthWestern College Pub; 7 edition, 2007. - 672 p.
21) Карплюк С.В., Коробейнікова Т.І., Савицька Л.А. Система обміну миттєвими повідомленнями в корпоративній мережі – Громадська організація «наука та освіта без кордонів», XX Міжнародна науково-

практична інтернет конференція «Актуальні досягнення сучасних наукових досліджень», 2019 – С. 19-25
https://ispic.ngo-seb.com/assets/files/20_conf_17.09.19_P.1.pdf
22) Alabama Computer Crime Act, Computer Crime Statutes, May 1, 1995. Excerpts from Computer Search Warrants.
23) Alabama Computer Crime Act, Computer Crime Statutes, May 1, 1995. Chapter 25 - Counterfeiting and Forgery.
24) B. Eckel Thinking in Java – NY, Prentice Hall, 2000
25) CHAPTER 21A - Privacy Protection Subchapter I-first Amendment Privacy Protection.
26) Code of Virginia, Title 18.2, Chap. 5 Crimes Against Property, Article 7.1. Computer Crimes.
27) Communications Privacy Act of 1986 Chapter 119. Wire and Electronic Communications Interception and Interception of Oral Communications.
28) Criminal Code of state Utah. http://www.utahcusa.criminal.codes.ng.htm
29) Federal Guidelines for Searching and Seizing Computers.
30) J. Shirazi Java Performance Tuning – Sebastopol, O'Reilly, 2000
31) P. C. Dibble Real-Time Java Platform Programming – NY, Prentice Hall, 2002
32) S. Stelting, O. Maassen Applied Java Patterns – NY, Prentice Hall, 2001
33) United States Code, TITLE 17 - January 1, 1998. CHAPTER 1 - SUBJECT MATTER AND SCOPE OF COPYRIGHT.
34) Вирт Н. "Алгоритмы + структуры данных = программы." Москва: Мир, 1977.
35) Жидецький В. Ц. Охорона праці користувачів комп'ютерів. Навчальний посібник. – Вид. 2-е, доп. – Львів: Афіша, 2001. – 176 с.
36) Єжова. Л.Ф. "Алгоритмізація та програмування процедур обробки інформації". Київ: КНЕУ, 2000р.
37) Митчелл К. Керман. "Программирование и отладка. Учебный курс." Москва-Киев: Вильямс, 2003.
38) Строуструп Б. "Справочное руководство по языку программирования JAVA с комментариями." Издательство "Мир", 1992г.
39) Семотюк В. „Програмування в середовищі JAVA ", Навч посібник для студентів технічних спеціальностей Львів 2000.

40) Хосс Джексон, Тод Маркес “ JAVA ” справочник професионала. Москва: СП ЭКОМ, 2003.
41) Уинер Р. " Язык JAVA" М.:Мир, 1991.
42) Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.

ДОДАТКИ