Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Біотехнічних систем
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Агентно-орієнтована система для поліклінічного відділення

Виконав(ла): студент(ка) 6 курсу, групи РБм-61
спеціальності 163 Біомедична інженерія
(шифр і назва спеціальності)

	(підпис)	Дороніна І.О. (прізвище та ініціали)
Керівник	(підпис)	Яворська Є.Б. (прізвище та ініціали)
Нормоконтроль	(підпис)	Паляниця Ю.Б. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Яворська Є.Б. (прізвище та ініціали)
Рецензент	(підпис)	Чайковський А.В. (прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Прикладних інформаційних технологій та електроінженерії
(повна назва факультету)

Кафедра Біотехнічних систем
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Яворська Є.Б.
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 163 Біомедична інженерія
(шифр і назва спеціальності)

студенту Дороніній Іванні Олексіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Агентно-орієнтована система для поліклінічного відділення

Керівник роботи Яворська Євгенія Богданівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*02*» *листопада* 2020 року № *4/7-793*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)

1. Аналітична частина
2. Основна частина
3. Науково-дослідна частина
4. Спеціальна частина
5. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	Зелінський І.М., доц. каф. ПВ		
безпека в надзвичайних			
ситуаціях	Стадник І.Я., проф. каф. ОХ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу 1		
4	Виконання розділу 2		
5	Виконання розділу 3		
6	Виконання розділу 4		
7	Виконання розділу 5		
8	Оформлення пояснювальної записки		
9	Оформлення графічного та презентаційного матеріалу		
10	Попередній захист		
11	Захист		

Студент _____ Дороніна І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дороніна Іванна Олексіївна. Агентно-орієнтована система для поліклінічного відділення. – Рукопис.

Кваліфікаційна робота магістра за спеціальністю 163 – біомедична інженерія, ТНТУ, Тернопіль, 2020.

Кваліфікаційна робота мігістра на тему «Агентно-орієнтована система для поліклінічного відділення» складається із вступу, аналітичної, основної та науково-дослідної частини, розділу «Охорона праці та безпека життєдіяльності», висновків, переліку літератури та додатків.

В аналітичній частині дано визначення програмного агента та проведений аналіз існуючих програмних аґентів. Розглянуто типи програмних аґентів, інструментарій для створення програмних аґентів. Проаналізовано програмні платформи для створення програмних аґентів. Приділено увагу мультиагентним системам та архітектурі мультиагентних застосувань.

Розглянуто питання застосування агент-орієнтованого моделювання в сфері охорони здоров'я.

Ключові слова: мультиагентні системи, програмний агент, медичний заклад, імітаціна модель.

ANNOTATION

Doronina Ivanna. Agent-oriented system for outpatient department. - Manuscript.

Master's qualification work in specialty 163 - biomedical engineering, TNTU, Ternopil, 2020.

The master's qualification work on "Agent-oriented system for the outpatient department" consists of an introduction, analytical, main and research part, section "Occupational safety and health", conclusions, bibliography and appendices.

The analytical part defines the software agent and analyzes the existing software agents. Types of software agents, tools for creating software agents are considered. Software platforms for creating software agents are analyzed. Attention is paid to multiagent systems and architecture of multiagent applications.

The application of agent-oriented modeling in the field of health care is considered.

Key words: multiagent systems, software agent, medical institution, imitation model.

ЗМІСТ

ВСТУП

Актуальність теми. Агент-орієнтоване моделювання – це обчислювальний підхід, при якому агенти з зазначеним набором характеристик взаємодіє між собою і з навколишнім середовищем відповідно до заздалегідь визначеними правилами.

Основним завданням системних мультиагентних мереж є прийняття колективного рішення посеред висунутих альтернатив. Групи окремих агентів часто висувають вимогу щодо єдиного вибору альтернативи та плану дій і вибору напрямку на випадоу зміни середовища чи системи.

Тому актуальним є питання розробки методів побудови та використання мультиагентних систем для медичних закладів як основи медичних інформаційних систем, а також розробки підходу до використання інформаційних технологій побудови та використання програмних агентів інтегрованих в сфері охорони здоров'я з метою пошуку та видобування нових даних та знань.

Мета і задачі дослідження. *Метою дослідження* є підвищення інформатизації медичних працівників щодо перебігу процесів у медичному закладі шляхом застосування мультиагентних програмних рішень

Досягнення цієї мети вимагає розв'язання таких задач:

– провести аналітичний огляд відомих рішень для вибору напряму досліджень;

– проаналізувати відомі програмні агенти в системі охорони здоров'я;

– дослідити інструментарій для створення програмних агентів в медичному закладі;

– дослідити мови програмування і програмні платформи для створення програмних агентів.

– розробити імітаціну модель ресурсів програмного агента.

Об'єкт дослідження: процес розробки методів побудови та використання мультиагентних систем для медичних закладів.

Предмет дослідження: мультиагентні системи.

Методи дослідження побудовано на базі теорії обчислювальних процесів, системного аналізу та теорії прийняття рішень для обґрунтування створення програмних агентів. Для програмної реалізації алгоритмів опрацювання використано агентно-орієнтований підхід.

Практичне значення одержаних результатів. Застосування агент-орієнтованого підходу може застосовуватися для моделювання та прогнозу лікувального та діагностичного процесів у медичних закладах.

Апробація результатів дослідження. Викладені в кваліфікаційній роботі результати доповідались і обговорювались на III Міжнародній студентській науково-технічній конференції «Природничі та гуманітарні науки. Актуальні питання» (м. Тернопіль, 2020 р).

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз вимог до програмних аґентів

1.1.1 Аналіз предметної області. Згідно з класичним визначенням, програмний агент - це програма-посередник. Ці посередники взаємодіють з користувачами або іншими програмами і сенсом цієї взаємодії являється виконання яких-небудь дій від імені користувача або іншої програми [1].

Можна дати і інше відносно вільне визначення програмного агента, в якому агент виступає самодостатньою програмою, яка здатна контролювати своє власне ухвалення рішень і дій, ґрунтуючись на власному сприйнятті свого оточення, і які спрямовані на реалізацію однієї або декількох цілей.

Якщо розглядати відмінності між програмним агентом і додатком, то можна сказати, що головна відмінність полягає в його поведінці.

З точки зору концепції, то агент забезпечує спосіб опису складної програмної суті, якій властива автономна дія виконання завдань від імені користувача. Встановлення агента здійснюється шляхом опису його поведінки [2].

Існує декілька підходів до визначення поняття програмного агента, але загальна ідея полягає в тому, що агенти більш автономні, ніж об'єкти. Агент має автономність в плані вибору завдань і пріоритетів, а так само агентові властива гнучка поведінка.

У завдання програмного агента входять:

1) самостійна робота і контроль своїх дій;
2) взаємодія з іншими агентами;
3) зміна поведінки залежно від стану зовнішнього середовища;
4) видача достовірної інформації про виконання заданої функції і тому подібне.

5) Мультиагентні системи - це системи, що складаються з автономних інтелектуальних аґентів, що взаємодіють один з одним і пасивного середовища, в якому агенти існують і на яку також можуть впливати.

Існують різні класифікації програмних аґентів. Розглянемо найбільш популярну класифікацію, де виділено 4 типи програмних аґентів [3]:

1) агенти-покупці чи т.з. торгівельні боти;
2) персональні агенти;
3) агенти з моніторингу та спостереження;
4) агенти по отримуванню та аналізу даних.

Але ця класифікація не зовсім актуальна на сьогоднішній момент, оскільки з моменту написання цієї класифікації багато що помінялося. Так наприклад, за класифікацією, агенти-покупці здійснюють моніторинг мережевих ресурсів для отримання даних щодо товарів та послуг. На сьогодні даний процес значно розширився за межі звичайних пошукових систем [4].

Прикладом сьогоднішнього торговельного бота виступає інструмент трейдера, він є спеціальним програмним забезпеченням, яке здійснює торгівлю на біржі автоматично, без допомоги людини. Торговельний бот працює за певними правилами, по своїй стратегії. Стратегій існує дуже багато і вони дуже різні. Боти також можуть працювати на комп'ютерах користувачів, в "хмарах" і як плагіни для торговельних систем.

Боти можуть вирішувати декілька основних завдань. Перша з них - це автоматизація рутинних операцій. Деякі стратегії торгівлі мають на увазі постійне виконання ордерів і облік великої кількості даних, що дуже скрутно робити вручну. Інші стратегії мають на увазі дуже швидке реагування на зміни на біржі і ухвалення простих, але швидких рішень. Боти також можуть використовуватися трейдером (людиною, торгуючою на біржі) як допоміжний інструмент для яких-небудь операцій.

Персональні агенти, за класифікацією– інтелектуальні агенти, які працюють від імені користувача. Прикладом таких дій є відправка даних, отримання і автоматична обробка даних.

Персональні агенти виконують наступне:

1) пошук інформації за визначеним запитом;

2) заповнення WEB-форм та збереження даних для використання;;

3) скан WEB-сторінки з метою пошуку та виділення важливої частини тексту;

4) можливість обговорення широкого спектру тематик;;

5) пошук роботи, відправка резуме тощо;

6) синхронізація соціальних мереж.

Агенти з моніторингу як правило проводять моніторинг складних комп'ютерних мереж і можуть стежити за конфігурацією кожного комп'ютера, підключеного до мережі.

До розряду особливих відносяться агенти, які моделюють процес прийняття рішень

Клас агентів по отриманню та аналізу даних відносять до торгівельних ботів.

1.2 Опис інструментарію для створення програмних аґентів

Для створення програмних аґентів використовують мови опису і реалізації, які зручно представляти у виді багатошарової структури, яка складається з п’яти шарів:

1) мови і програмні засоби реалізації аґентів;

2) мови комунікації аґентів;

3) мови опису поведінки аґентів і законів середовища;

4) мови представлення і управління знаннями;

5) мови формалізації і специфікації аґентів і мультиагентних систем (МАС).

Засобами взаємодії агентів є мови комунікації аґентів (*FIPAACL*, *KQML*), які забезпечують обмін знаннями та інформацією між ними. *FIPAACL* на відміну від засобів *RPC*, *RMI*, *CORBA*, які використовуються для обміну інформацією між додатками та мають складу семантику. Перевагами *FIPAACL* є:

- керування судженнями, правилами, діями;

- за допомогою повідомлення здійснюється опис очікуваного стану, а не процедури чи методу.

Хоча *ACL* не охоплює всі об'єкти якими можуть здійснювати обмін агенти (плани, цілі, досвід, стратегії тощо). За рахунок використання *ACL* (на технічному рівні) агенти передають повідомлення по мережі із використанням протоколів нижчого рівня (наприклад, SMTP, TCP/IP, POP3, чи HTTP).

Із використанням мови взаємодії агентів здійснюється передача інформації між різними агентами. Існує два підходи при проєктуванні мов взаємодії:

1. Процедурний – містить обмін процедурними директивами / командами та реалізується за допомогою мов програмування *Java* або TCL.

2. Декларативний – використовуються для зв'язку декларативні інструкції, типу визначень, припущень, знань тощо.

Оскільки, часто існують обмеження на процедурні підходи (наприклад, складність при координуванні сценаріїв), то, для створення мов взаємодії агентів, перевага надається декларативним мовам. Найбільш застосовуваними є такі мови як *KQML* із своїми діалектами та *FIPAACL*.

У контексті практичної побудови аґентів і МАС основну роль відіграють інструментальні засоби програмування та комунікації агентів. Для забезпечення узгодженої взаємодії агентів використовують мови комунікації (*ACL*, *KQML*) та координації аґентів (AGENTALK). Ці мови можна розглядати як багаторівневі структури, що включають рівень представлення знань, рівень переговорів або координації, рівень стратегій комунікації, і тому подібне. Так, мова *KQML* (*Knowledge Query and Manipulation Language*) – службова і використовується для підтримування взаємодії агентів у розподілених застосуваннях, опирається на спецпротокол перенесення знань SKTP (*Simple Knowledge Transfer Protocol*). У свою чергу, мова *ACL* (*Agent Communication Language*), що претендує на роль стандарту для комунікації аґентів, складається з трьох частин- словника, "внутрішньої мови" KIF (*Knowledge Interchange Format*) і "зовнішньої мови" *KQML*. Повідомлення, що передається на мові *ACL*, може трактуватися як *KQML*-вислів, «аргументами» якого виступають пропозиції у форматі KIF, побудовані з елементів словника *ACL* [Genesereth and Ketchpel, 1994].

Розглядають три класи програмних засобів, які використовують для реалізації агентів:

1) об’єктно-орієнтовані мови програмування (як правило C++, *Java*);

2) бібліотеки аґентів;

3) середовища розробки аґентів.

Посеред відомих бібліотек агентів виділяють: 1) бібліотека інтелектуальних агентів Intelligent Agent Library (розробник BITS & PIXELS) – підтримує розробку мобільних аґентів, призначена для забезпечення комунікації аґентів і побудови їх груп, базується на використанні мови *KQML*, має ілюстровані приклади агентів функціонуючих у WEB-додатках; 2) система KAISA (розробник FUJITSU); та 3) AGENTX (розробник International Knowledge Systems) – бібліотека розподілених обчислень в середовищі *Java*.

Одним з найвідоміших і вже таких, що зарекомендували себе інтегрованих середовищ для розробки інтелектуальних програмних аґентів є AgentBuilder фірми Reticular Systems, Inc. Цей засіб складається з двох основних компонентів: інструментарію (TOOLKIT) і виконавської системи. Інструментарій включає :

1) засоби управління процесом розробки програмного забезпечення, заснованого на агентах;

2) засоби аналізу області функціонування агента;

3) засоби проектування і розробки мереж з взаємодіючих аґентів;

4) засоби моделювання поведінки окремих аґентів;

5) засоби відладки і тестування програмних аґентів.

Виконавська система містить машину агента (AgentEngine), формуюче середовище для реалізації програмних аґентів. Агенти, розроблені за допомогою середовища AgentBuilder, взаємодіють на мові *KQML*, в основі якого лежать примітивні дії – перформативи. Призначення інструментального засобу AgentBuilder полягає у наданні розробнику програмного забезпечення, яке використовує агентів, інтегрованого середовища для швидкого створення інтелектуальних агентів та написання складних програм на їх основі. У таблиці 1.1 наведені приклади засобів для розробки аґентів.

Провівши аналіз (див. табл. 1.1, 1.2) встановлено, що засоби, які використовуються для розроблення агентів поділяються на дві групи: 1) побудовані із використанням *Java* при розробленні мережних застосувань на основі мобільних агентів і взаємодіють через протокол TCP/IP, та 2) побудовані на базі інших мов.

Таблиця 1.1

Приклади засобів розробки агентів

Назва мови (засоби розробки)	Коротка характеристика / призначення
AGENTALK	Засіб для розробки автономних мобільних агентів на базі мови сценарію TCL з підтримкою TK - інструментарію для створення графічного інтерфейсу
AGENTTOOL	Агентна структура на базі Java. Використання GUI для конструювання системи
FIPA - OS	Реалізація елементів, що містилися в FIPA -специфікації по взаємодії агентів. Java -реалізація архітектури для розробки агентів
ECHELON	Засіб для розробки агентів в мережі LonWorks
JAFMAS	Програма, що забезпечує створення мультиагентних системи на Java. Підтримує міжагентне спілкування на KQML
Remembrance Agents	Засіб для розробки агентів, що спостерігають за користувачем і пропонують інформацію, необхідну в даний момент
UMMON	Засіб для створення самонавчального агента. Містить методи ШІ для досягнення "людського" спілкування
Virtual Secretary Project (ViSe)	Створення моделі користувача на основі інтелектуальних агентів для виконання секретарських завдань (технологія TCL/TCLX/TIX/TK)

Таблиця 1.2

Порівняння параметрів інструментальних засобів розробки агентів

Параметр	Найменування інструментальних засобів					
	JAFMAS	Agen Talk	Agent Tcl	Telescript	Swarm	Echelon
Мова розробки	Java	Lisp	Tcl	Telescript	Objective C	Silicon chip
Підтримка ОС	Усе	UNIX	UNIX	UNIX	UNIX	Echlon chip
Реалізація агента	програмна	програмна	програмна	програмна	програмна	апаратна
об'єктно-орієнтоване програмування	+	-	-	+	+	+
Наявність у агента свого плану дій	+	-	+	+	+	+
Комунікаційний протокол	TCP/IP і UDP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP
Мобільність агентів	+	-	-	-	-	-

1.3 Мови програмування і програмні платформи створення програмних агентів

При програмуванні аґентів використовують наступні мови програмування: універсальні, представлення знань, переговорів і обміну знаннями, сценаріїв, спеціалізовані, символьні та мови логічного програмування.

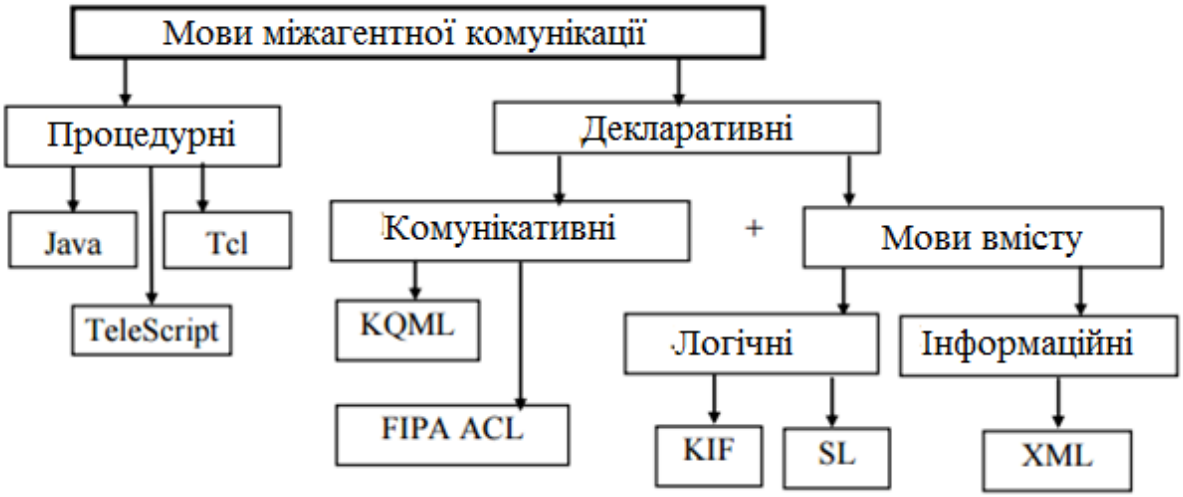

Рис. 1.1 – Мови міжагентних комунікацій

Враховуючи той факт, що *Java* є незалежною мовою, яка компілює свій код у машинний, додатки, розроблені за допомогою засобів цієї групи, можуть працювати на інших платформах. Відмінність між засобами полягає у тому, що не всі підтримують мовний обмін повідомленнями із використанням *KQML*. Недоліком є відсутність класів, які визначають соціальну поведінку аґентів.

Інструментальні засоби іншої групи, які в більшості випадків, призначаються, коли проектують складні динамічні агентні структури та реалізовують середовища для проектування складних динамічних агентних структур, також є структури для створення мобільних застосувань (*Telescript*, *AgentTCL*). У такому випадку агенти розробляються відповідно BDI-моделі. Комунікації відбуваються із використанням протоколу TCP/IP. Проте такі засоби мають слабкі можливості за погодженням і співпрацею між агентами і працюють на обмеженій кількості платформ, оскільки мова програмування не є уніфікованою.

Більшість комунікативних агентних мов грунтуються на мовній взаємодії із використанням стандартних ключових слів.

Існує два підходи щодо розроблення мов – процедурний і декларативний. При процедурному – комунікації проходять під час виконання інструкцій, проектування мови здійснюється із використанням *Java* чи *TCL* (*Tool Command Lanuage*). При декларативному – реалізація комунікацій відбувається із використанням опису. Декларативний підхід набув великого поширення при створенні мов спілкування агентів, поширеною є *KQML*, яка використовується як мова взаємодії у різного роду МАС та середовищах для їх програмування.

При розробленні поміжагентного повідомлення необхідною є мова для представлення власне змісту. З цією метою використовують «логічні мови» – представлення знань як логічних виразів (одним із представників є *Knowledge Interchange Format – KIF*) та «інформаційні мови» – опис типів інформаційних елементів за встановленими правилами. *KIF* використовується для спрощення обміну знаннями у системах штучного інтелекту, синтаксично побудований із використанням *CommonLISP*. Формат *KIF* є декларативною мовою, яка дозволяє різним системам обмінюватися онтологіями використовуючи тим самим в роботі обчислювальні переваги цих систем.

Друга логічна мова змісту – SL (Semantics Language), запропонований FIPA. SL – пропозиції виражаються логікою ментальних стосунків і дій. Ментальна модель агента заснована на представленні трьох примітивів : переконання, невизначеність, вибір. Основну властивість SL – логіки дозволяють змодельованим агентам знаходиться відповідно до їх ментальних відносин.

Для програмної реалізації МАС використовуються програмні системи з набором засобів як для програмного опису діяльності аґентів і стану середовища, так і для контролю і управління процесом їх взаємодії і роботи. Такі системи називаються агентними платформами.

Існує досить великий набір платформ, відповідних для створення мультиагентних систем, і цей набір постійно поповнюється (NetLogo, StarLogo, Repast Simphony, Eclipse AMP, JADE, Jason). Самі ці платформи реалізовані дуже по-різному: від окремих середовищ розробки до вбудовуваних плагінів і

бібліотек, що підключаються. Вони можуть використовувати як вже існуючі мови різних парадигм, так і мови, спеціально розроблені для побудови програмних аґентів, наприклад, AgentSpeak в системі розробки Jason.

За об'ємом інструментарію платформи можна розділити на простих і складних. Прості (NetLogo, StarLogo) мають маленький, але потужний інструментарій, що дозволяє швидко писати досить складні програми. Якщо для формалізації і налаштування моделі ще підходять прості системи, то для реалізації краще вибрати складну, яка надає більше можливостей, хоча ними і важче користуватися. Розглянемо деякі агентные платформи.

Платформа розробки *Java* – аґентів *JADE* (*Java Agent Development Framework*) використовується для створення мультиагентних систем і додатків відповідно до стандартів FIPA [5] для інтелектуальних аґентів. Вона включає середовище виконання аґентів. Агенти реєструються і працюють під його управлінням, в ній передбачені механізми створення, видалення, активація аґентів (див. рис. 1.2).

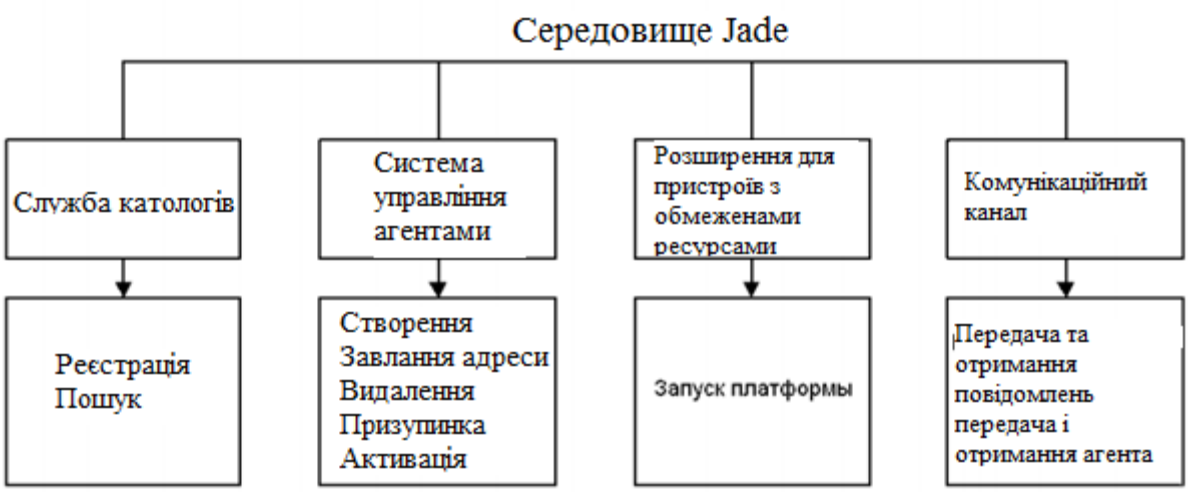

Рис.1.2 –Компоненти *JADE*

У середовищі є система управління агентами (*Agent Management System*), і служба каталогів (*Directory Facilitator*). Також, *JADE* має бібліотеку класів для розробки аґентів і агентної системи та набір графутиліт для адміністрування і спостереження за життєдіяльністю аґентів, що функціонують в системі.

Важливою особливістю *JADE* є підтримка виконання аґентів на пристроях з обмеженими можливостями *CLDC* (*Connected Limited Device Configuration*), наприклад, мобільних телефонах, кишенькових комп'ютерах (PocketPC, Palm та ін.), за рахунок використання розширення *LEAP* (*Lightweight Extensible Agent Platform*).

Розглянемо самостійне середовище розробки *NetLogo*. Ця система є типовим прикладом автономного середовища розробки МАС. До основних плюсів цієї платформи відноситься великий об'єм засобів взаємодії, що надаються, з користувачем – різних кнопок і регуляторів для вводу і коригування інформації, а так само вільне розповсюдження системи.

Крім того, ця платформа надає, разом із стандартною збіркою *NetLogo*, приклади моделей і систем, які надають допомогу для розуміння облаштування даної агентної платформи та її можливостей. Але технічні можливості системи є досить обмеженими: 1) набір засобів візуалізації є строго фіксований і регламентований, крім того, відсутня можливість додати своє; 2) система не дає можливості структурувати програму, програма пишеться одним файлом; 3) специфіка мови програмування.

NetLogo це скриптова мова, а це в свою чергу впливає на швидкодію програми. У системі відсутні інструменти інструменти для розподіленого, незалежного виконання, а у функціонуючих в ній суб’єктів немає структури агента, і вони зберігають тільки власний стан. Їх поведінка реалізується виконанням функцій на класі об'єктів, тобто, по суті, у рамках об'єктної парадигми програмування. З іншого боку, така простота взаємодії знаходиться в гармонії із слабкими можливостями структуризації програми і взагалі організації процесу розробки. Саме вона дозволяє писати невеликі, не громіздкі і не засмічені зайвими і непотрібними практично (хоча і важливими теоретично і структурно) елементами, але досить потужні програми. Цьому також сприяє і відмічена вище простота роботи з широким, хоча і жорстко обмеженим набором засобів візуалізації.

Розглянемо платформи з окремими модулями, що підключаються, для розробки МАС. Окремі модулі не є ще власне бібліотеки, набори виключно

засобів розробників для написання системи, оскільки вони надають широкий набір методів взаємодії з користувачем. Проте вони вже і не є фактично автономними системами, а, у вигляді модулів, підключаються, до вже наявних засобів розробки. Це дозволяє поєднувати стандартні засоби розробки із спеціалізованими і призначеними виключно для агентного моделювання засобами. До цього класу інструментів відносяться системи Jason, Eclipse AMP.

Одним із представників даного класу систем є Repast Simphony – модуль, який підключається, до системи розробки (IDE) Eclipse. По-перше, така реалізація дозволяє використовувати існуючі можливості середовища, та і нових мов програмування учити не доводиться. А по-друге, методи розробки агентних систем, що надаються, дозволяють сконцентруватися саме на написанні аґентів, причому використовувати при цьому дуже наочні і цікаві засоби.

Головним плюсом, безумовно, являється безпосередня робота з агентами і системами їх взаємодії. Так, існують прості методи створення агента, завдання його параметрів, опис його поведінки навіть з використанням блок-схеми. Усе це робить роботу з системою досить зручною і зрозумілою.

Проте, не усе так добре, як може здатися спочатку. Окрім чисто випадкових мінусів, недоліків в середовищі взаємодії з користувачем (неможливість відкату при здійсненні деяких важливих 30 операцій, складнощі при збереженні і відновленні і тому подібне) і часткової недопрацьованої системи (часом з'являються помилки, що перешкоджають роботі системи, відстежити які може тільки розробник), є ще і труднощі, мабуть, властиві такому способу рішення в цілому. А саме – громіздкість і частенько зайвий об'єм роботи, що проробляється.

Крім того, незважаючи на уявну легкість створення програм в такому середовищі, сам процес створення є строго регламентованим, і якщо щось робиться не так, як спочатку задумано творцями системи, це може привести до помилки на нижчому рівні, тобто на рівні вже власне середовища розробки і мови програмування.

Усе це робить необхідним або періодичне використання низькорівневих засобів розробки, або строге наслідування схем побудови систем. Таке положення

частково перекреслює ту величезну вигоду, яку приносить поєднання стандартних і потужних засобів розробки із спеціалізованими методами.

1.4 Засоби специфікацій типових моделей

Розглянемо типову модель, яка дістала назву Reticular Agent Mental Model (RAMM) і є розвитком моделі Шохама (Shoham), де усі дії виконуються тільки як результат певних зобов'язань. У рамках RAMM ця ідея розширена до рівня загальних правил поведінка, яка визначає причину дії агента в кожній точці його функціонування. При цьому правила поведінки фіксують безліч можливих "відгуків" агента на поточний стан середовища так, як це наказує полаганиями.

Схематично процес проектування та реалізації агент-орієнтованих застосувань базованого на AgentBuilder ToolKit показано на рисунку 1.3.

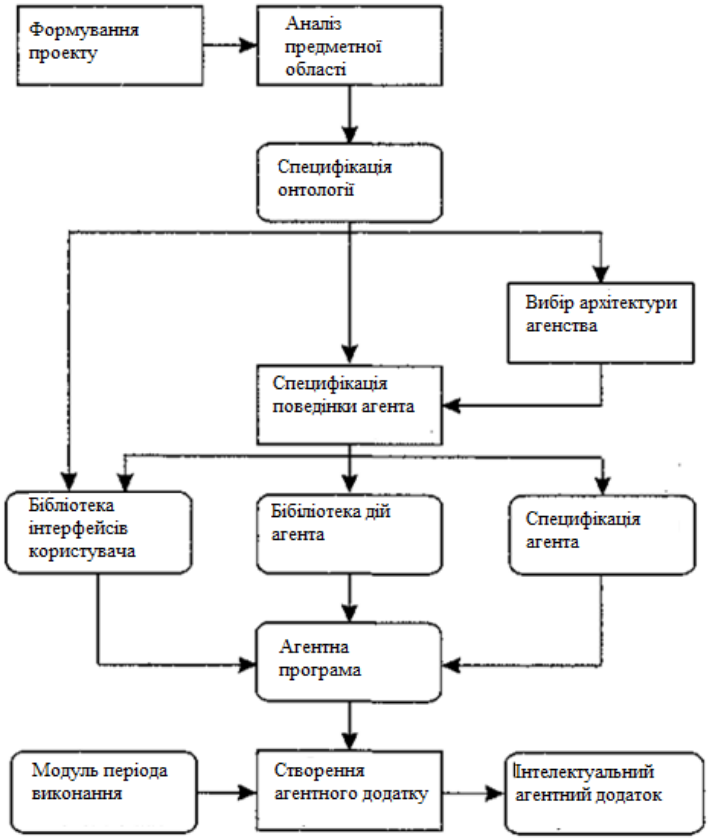

Рис. 1.3 – Типова схема процесу розробки агентно-орієнтованих застосувань, які базуються на базі AgentBuilder ToolKit

Для специфікації поведінки аґентів в системі AgentBuilder використовується спеціальна об'єктно-орієнтована мова RADL (Reticular Agent Definition Language). Правила поведінки в цій мові можуть розглядатися як конструкції виду WHEN - IF - THEN.

WHEN - частина правила адресована новим подіям, що виникають в оточенні агента і включає нові повідомлення, отримані від інших аґентів.

IF - частина порівнює поточну ментальну модель з умовами застосовності правила. Зразки в IF - частини працюють на намірах, зобов'язаннях і можливостях, визначених в ментальній моделі.

THEN - частина визначає дії у відповідь на поточні події і стани ментальної моделі і зовнішнього оточення. Вони можуть включати оновлення ментальної моделі, комунікативні і внутрішні дії.

На рисунку 1.4 представлена модель "життєвого циклу" агента в системі AgentBuilder.

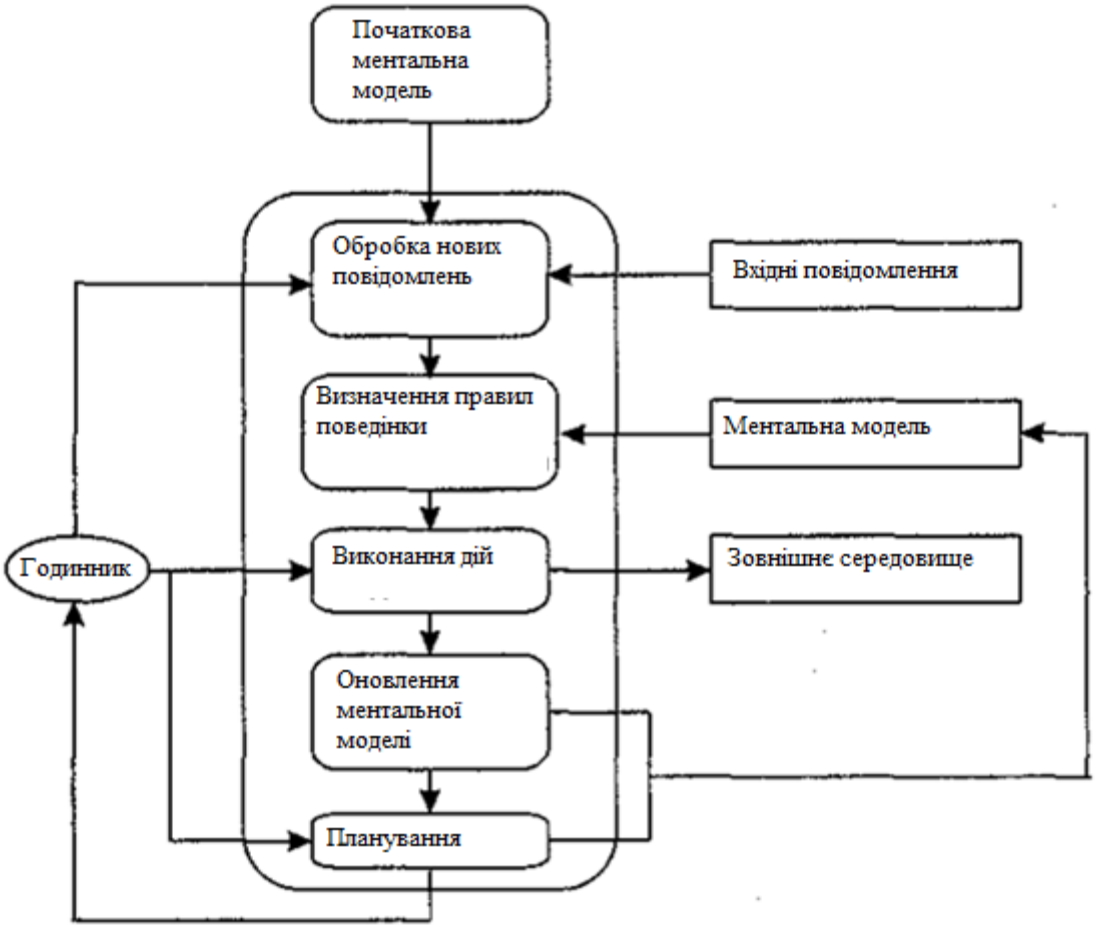

Рис. 1.4 – Модель "життєвого циклу" агента в системі AgentBuilder

У мові RADL активно використовуються зразки, є досить розвинені засоби роботи із змінними і показний набір дій, що включає формування перформативів мови KQML. Структури даних, на яких "працює" ця мова, являються, по суті, фреймами, а самі правила - суть продукції спеціального виду. Мова підтримана на інструментальному рівні системою спеціальних мовно-орієнтованих редакторів.

Специфікація поведінки аґентів і їх ментальних моделей складає спеціальний файл (agent definition file), який використовується спільно з класами і методами з бібліотеки дій аґентів і бібліотеки інтерфейсів. Цей файл інтерпретується у рамках компонента Reticular's Run - Time Agent Engine, оточення періоду виконання AgentBuilder, що є частиною.

Оцінюючи підхід до специфікації моделей поведінки аґентів, використовуваний в AgentBuilder, можна констатувати, що в цілому це серйозна система представлення і маніпулювання знаннями, орієнтована на опис моделей типу RAMM. В той же час в цій моделі відсутні засоби експліцитного управління виводом, які могли б істотно збільшити функціональну потужність мови. Немає в моделі і засобів явної фіксації стану агента, відмінних від прапорів і/або значень змінних. Не цілком ясно і те, як в специфікації моделей поведінки можуть бути враховані різні, але одночасно співіснуючі "лінії поведінки", що характерно для дійсно інтелектуальних аґентів. Не цілком обґрунтованим представляється і використання режиму інтерпретації для реалізації поведінки аґентів.

Але в цілому можна ще раз відмітити, що інструментарій AgentBuilder є сучасним і потужним засобом проектування і реалізації МАС.

1.5 Архітектура мультиагентних застосувань

Потрібно враховувати, що середовище, в якому діє агент, володіє певними особливостями і залежить від інформації, наявної у агента, а також від властивостей: дискретність стану, детермінована дій, динамічність або статичність, синхронна або асинхронна зміна стану і тому подібне.

На відміну від традиційних систем, в яких рішення шукається за допомогою централізованих, послідовних і детермінованих алгоритмів, в МАС рішення

досягається в результаті розподіленої взаємодії множини агентів – автономних програмних об'єктів, націлених на пошук можливо не стільки повністю оптимального, скільки найкращого з можливих рішень на кожен момент часу. Якщо знайдений агентом кращий варіант вже забронюваний іншим агентом, агенти виявляються здатні виявити конфлікт і дозволити його шляхом переговорів, в ході яких досягається компроміс, що відбиває тимчасове, і, як правило, нестійка рівновага (баланс) їх інтересів.

МАС або агентно-орієнтовані програмування є наступним кроком в розвитку об'єктно-орієнтованого програмування (ООП) і інтегрують в собі досягнення останніх десятиліть у сфері штучного інтелекту, паралельних обчислень і телекомунікацій.

Розрізняють три базові типи архітектури МАС :

1) архітектура, заснована на принципах і методах роботи зі знаннями;

2) архітектура, заснована на поведінкових моделях;

3) гібридні побудови, що поєднують в різних співвідношеннях особливості двох перших типів.

У МАС із першою архітектурою ухвалення рішень про дії агента відбуваються на основі логічних чи псевдологічних міркувань. Такий агент розглядається як спеціальний випадок базованої на знаннях системи.

Спершу така ідея будувалася на чисто логічній основі і представлялася дуже перспективною. Пізніше встановлено, що було виявлено, що числення предикатів І-го порядку, яке лежить в основі такого підходу нерозв'язне. Як варіант вирішення проблеми розроблено архітектуру *Belief Desire Intention* (BDI) [6].

У системах, побудованих на поведінкових моделях, головним поняттям, що утворює, є "ситуація", в якій виявився агент, і "спонукальний мотив" до вироблення адекватного цій ситуації дії або поведінки ("реакція") агента. Поведінкові моделі найчастіше будуються на правилах вироблення адекватних реакцій. Чи, якщо різноманітність ситуацій не катастрофічна - застосовуються алгоритми, що дозволяють створити кінцевий автомат з обмеженими можливостями. Агенти МАС цього типу називаються реактивними, а їх реакція на

той або інший спонукальний мотив завжди розглядається як результат співвідношення стану зовнішнього середовища і внутрішнього стану агента.

До перспективниїх відносяться гібридні МАС, які об’єднують якості поведінкових та продукційних моделей. Вони часто використовують принцип спеціалізації своїх аґентів. Примітно, що у БЗ аґентів цих систем складається із знань 3-ьох рівнів – предметна область, взаємодія з іншими агентами, знання для оптимального керування системою.

1.6 Висновок до розділу 1

У розділі дано визначення програмного агента та проведений аналіз існуючих програмних аґентів. Розглянуто типи програмних аґентів. Розглянуто інструментарій для створення програмних аґентів, який включає мови і програмні засоби реалізації аґентів, мови комунікації аґентів, мови опису поведінки аґентів і законів середовища, мови представлення і управління знаннями, мови формалізації і специфікації аґентів і мультиагентних систем (МАС).

На основі аналізу відомих мов програмування програмних аґентів, встановлено, що найчастіше застосовуються мови ООП (C++, *Java*), рідше символьні і мови логічного програмування (LISP, Oz). Проаналізовано програмні платформи для створення програмних аґентів: NetLogo, StarLogo, Repast Simphony, Eclipse AMP, JADE, Jason.

В результаті цього аналізу зроблено висновок, що найбільш прийнятна платформа JADE, оскільки єдиним істотним мінусом при такому підході є необхідність детального вивчення бібліотеки. Проте використання самостійних систем не дасть нам в цьому особливої переваги, оскільки теж вимагає детального вивчення особливої мови програмування, притому досить небагатого по функціональності, що надається, і розширюваності.

Розглянуто засоби специфікацій типових моделей на прикладі спеціальної об'єктно-орієнтованої мови *RADL* (*Reticular Agent Definition Language*).

Приділено увагу мультиагентним системам та архітектурі мультиагентних застосувань. В результаті виконаної роботи можна зробити висновок, що

мультиагентні системи - це один з нових перспективних напрямів штучного інтелекту. Ключовим елементом цих систем стає програмний агент, здатний сприймати ситуацію, приймати рішення і взаємодіяти з іншими агентами. Ці можливості радикально відрізняють МАС від існуючих "жорстко" організованих систем, забезпечуючи ним таку принципово важливу нову властивість, як здатність до самоорганізації.

РОЗДІЛ 2
ОСНОВНА ЧАСТИНА

Агент-орієнтоване моделювання - це обчислювальний підхід, при якому агенти з зазначеним набором характеристик взаємодіє між собою і з навколишнім середовищем відповідно до заздалегідь визначеними правилами.

2.1 Порівняльний огляд агент-орієнтованих моделей (АОМ) в охороні здоров'я

Агент-орієнтоване моделювання являє собою висхідний підходом, при якому поведінка на мікрорівні породжує динаміку в макрорівні. На 1 представлена схема гіпотетичної агент-орієнтованої моделі охорони здоров'я, де індивіди, можуть мати різні характеристики індивідуального рівня - від ендогенних факторів до соціально-економічного статусу (синя верхня таблиця), а також характеристики на рівні спільноти (зелена верхня таблиця), які перетинаються разом, щоб формувати індивідуальну поведінку щодо здоров'я і використання медичних послуг. Агент-орієнтована модель в сфері охорони здоров'я може також явно включати ефекти від поточних процесів, такі як старіння і рух між групами (помаранчеві кола в середині схеми). В сукупності і біологічні, і поведінкові і соціальні процеси становлять систему охорони здоров'я.

Іншими відмінними властивості АОМ є автономність, гетерогенність і стохастичність. Автономність має на увазі, що агенти приймають рішення про те, як діяти, враховуючи їх поточні обставини і запрограмовані правила поведінки. Гетерогенність відбивається у відмінностях серед агентів і їх середовищем проживання, які можуть мати кілька статичних і тимчасових характеристик. Зміни в агента і характеристиках навколишнього середовища можуть відбутися несподівано з плином часу, в результаті чого зміниться хід майбутніх подій. Стохастичність дозволяє моделі розгортатися з часткою випадковості, що впливає

на поведінку і зміни в моделі. В результаті цих властивостей АОМ можуть використовуватися для розгляду нелінійних відносин під впливом декількох рівнів і міжособистісних взаємодій.

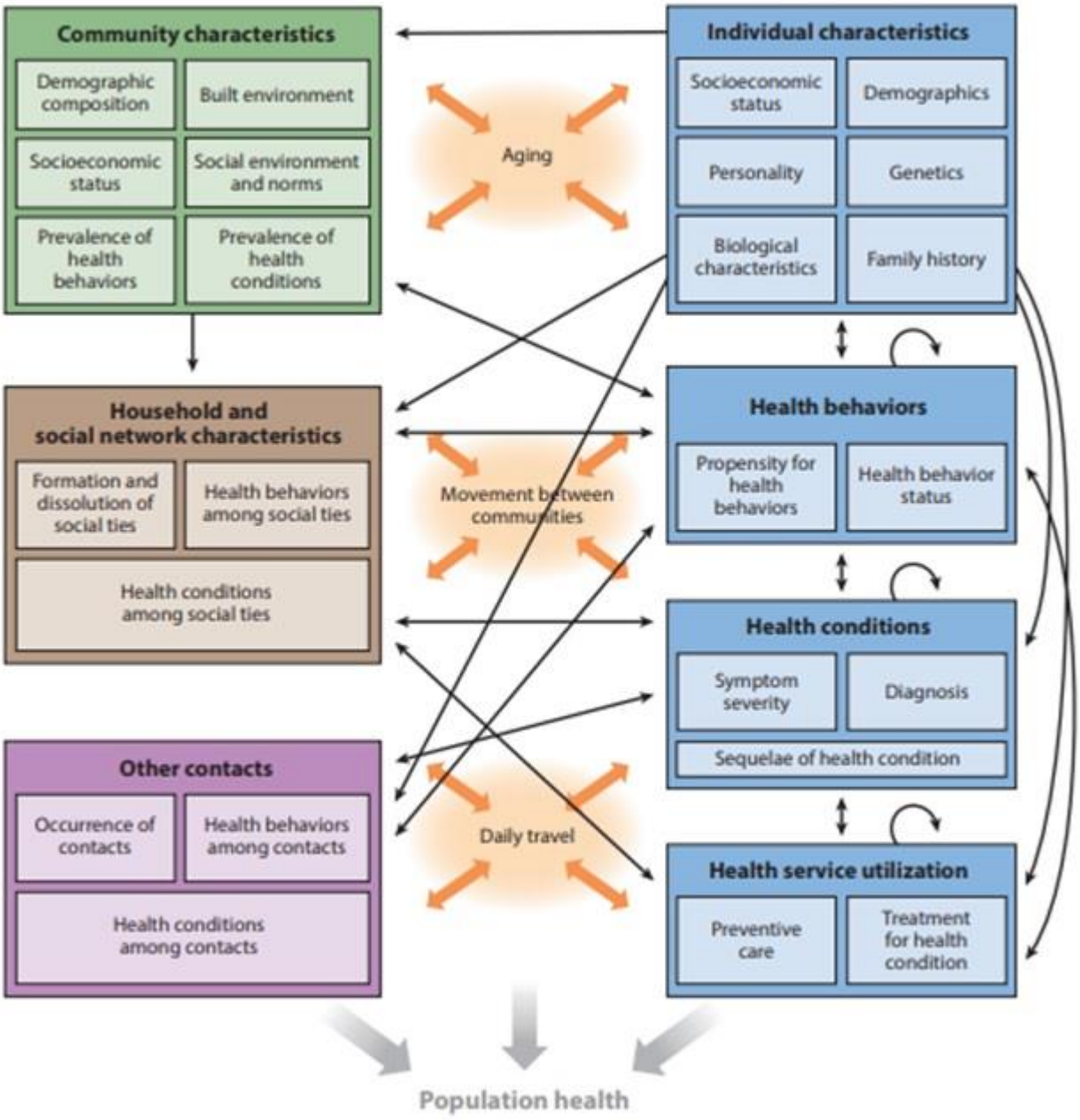

Рис. 2.1 – Схема моделювання взаємодій в сфері охорони здоров'я

2.1.1 Застосування агент-орієнтованих моделей в боротьбі з інфекційними захворюваннями. У громадській охороні здоров'я Агентне моделювання

історично використовувалося майже виключно для моделювання поширення інфекційних захворювань серед населення. АОМ є природним підходом для моделювання передачі інфекційного захворювання, тому що взаємодія між людьми часто призводять до поширеності інфекційних захворювань серед населення. Однак за останні 15 років ці методи все частіше застосовуються і до неінфекційних захворювань, включаючи поведінку щодо здоров'я, соціальної епідеміологія і до інших питань здоров'я населення, яке не пов'язане з традиційними інфекційними процесами.

2.1.2 Агент-орієнтовані моделі в питаннях інфекційної епідеміології. Застосування АОМ в інфекційній епідеміології при передачі інфекційних захворювань залежить від структури вразливе-інфікованих-видужали (SIR), запропонованої Kermack і McKendrick в 1920-х роках, в якій потоки між сприйнятливими, зараженими і відновленими агентами регулюються диференціальнимирівняннями. Розширення таких моделей SIR були використані в епідеміологічних центрах і лікарнях при прогнозуванні потоків пацієнтів.

Таким чином, до теперішнього часу Агентне моделювання інфекційних захворювань стало застосовуватися в плануванні громадської охорони здоров'я. Крім того, багато хто з розроблених можливостей моделювання, розширені і уточнені за допомогою програм по боротьбі з інфекціями, таких як MIDAS, і можуть застосовуватися до проблем суспільної охорони здоров'я, крім інфекційних захворювань.

2.1.3 Застосування агент-орієнтованих моделей в боротьбі з неінфекційними захворюваннями. Визнання того, що взаємозв'язки між має значення і для поширення неінфекційних захворювань призвело до розширення застосування АОМ в цій області. Проблеми ожиріння і йому подібні неінфекційні захворювання були предметом безлічі досліджень, враховуючи масштаби ожиріння. З'ясувалося, що дані захворювання є як проблемами суспільної охорони здоров'я, так і перебувають під комплексним впливом біологічних, поведінкових, соціальних і екологічних факторів ризику. Агентні моделі, які прогнозують

ожиріння серед населення, враховували соціальні контакти агентів, включаючи моделювання впливу місця проживання на масу тіла і сусідство з іншими національностями.

АОМ також використовується для вивчення діабету, висвітлюючи поширення діабету і зв'язну з ним втрату зору.

2.1.4 Застосування агент-орієнтованого підходу для моделювання шкідливих звичок населення. Застосування АОМ для моделювання здоров'я населення. Так, АОМ використовуються для прогнозування поширення шкідливих звичний серед населення, таких як алкоголізм, куріння, відсутність фізичного навантаження і вживання нездорової їжі. Агент-орієнтовані моделі підкреслили роль соціального впливу на політику боротьби проти тютюну і куріння. Також, сучасні агентні моделі куріння досліджували наслідки переходів на електронні сигарети щодо поширеності куріння серед населення і роль соціально-економічного статусу на куріння. Таким чином, проблеми здоров'я населення можуть бути розглянуті в динаміці щодо шкідливих звичок саме за допомогою агент-орієнтованого підходу.

2.1.5 Застосування агент-орієнтованого підходу для моделювання соціального здоров'я населення. Агентні моделі, які вивчають мережеві взаємодії в соціумі, особливо добре підходять для вивчення питань, що становлять інтерес для соціальної епідеміології, наприклад колективна поведінка, розподіл ресурсів та інші соціальні взаємодії, які є основними причинами захворювань. Так, автори змоделювали модель соціального насильства серед населення і перевірили альтернативні стратегії для зменшення насильства і його наслідків. Зокрема, було створено віртуальне уявлення дорослого населення Нью-Йорка, розподіленого по районам Нью-Йорка. Жорстокі переживання серед індивідів в моделі визначалися взаємодії з іншими агентами, соціо-демографічними характеристиками, симптомами психічного здоров'я, минулими фактами насильства і сусідства і зверненнями до співробітників поліції. У моделі проводилися різні експерименти зі зміни кооперації і взаємодії сусідів, які могли застерегти один одного від фактів

насильства. Вивчалися різні сценарії расового, соціальної і економічної нерівності серед сусідів. Так, результатом моделювання стало те, що при більшій кооперації сусідів, насильство скорочується. Однак в районах з етнічним нерівністю рівень насильства залишався незмінним.

2.2 Висновок до розділу 2

Агент-орієнтоване моделювання виходить все більшу поширеність в сфері охорони здоров'я. Якщо раніше вчені проводили моделювання тільки епідеміологічних захворювань і їх поширення серед населений за допомогою агентного підходу, то в даний час сфера застосування агент-орієнтованих моделей стала ширше. Так, агент-орієнтованих підхід може застосовуватися для моделювання поширення неінфекційних моделей шляхом соціальних контактів, для прогнозування поширення шкідливих звичок населення, а також для моделювання соціально-значущих проблем в суспільстві, таких як насильство.

РОЗДІЛ 3
НАУКОВО-ДОСЛІДНА ЧАСТИНА

3.1 Вступні завваги

Особливість завдань управління лікувальним процесом у медичному закладі полягає в тому, що в більшості закладах відсутнє єдине центральне оперативне управління: відділення та підрозділи працюють та планують свою діяльність самостійно, при взаємодії з іншими підрозділами. Крім того, тривалість виконання більшості лікувальних процедур (операцій, консультацій лікарів) неможливо визначити. У зв'язку з цим в останній час для оперативного планування та управління лікувальним процесом пропонуються методи, засновані на ідеях так званих мультиагентних систем. Кожна «сторона», яка задіяна у лікувальному процесі та представлена в оперативному плані (розкладі): лікарі, пацієнти, ресурси, представляється своїм програмним блоком – агентом чи багатьма агентами, які беруть участь у складанні розкладу «від свого імені», що відбувається з власних переваг і обмежень. Правила взаємодії агентів задаються таким чином, щоб отримати в результаті оптимальний розклад.

Є три типи робіт по застосуванню мультиагентних систем у медичних закладах. У роботах першого типу свідомо абстрагуються від конкретних особливостей лікувального процесу у медичних закладах різних типів. Метою цих робіт є – на узагальнених комп'ютерних моделях дослідити ефективність різних механізмів, що забезпечують взаємодію агентів, різних евристичних правил, складання розкладу і т.д.

У роботах другого типу кінцевою метою є застосування мультиагентної системи для оперативного планування лікувального процесу в реальних медичних закладах. Очевидно, що мультиагентна система не може працювати в автоматичному режимі, вона розглядається як інтерактивне середовище, яке дозволяє працювати в реальному часі. Постановка діагнозу, визначення стану пацієнта, вибір характеру медичного втручання та саме втручання залишається за

лікарями. Агенти беруть на себе формальну роботу, не вимагають медичної кваліфікації: здійснюють моніторинг, виконують деякі організаційні функції та складають розклад процесу лікування, мінімізують часу лікування, простій лікарів та обладнання.

У роботах третього типу мультиагентна структура використовується для побудови імітаційної моделі конкретної лікарні. У такі моделі з максимально можливою точністю відтворюється організаційна структура медичного закладу (або його частини), відповідні ресурси (палати, обладнання, персонал), механізм взаємодії підрозділів у ході лікувального процесу та реальні статистичні характеристики потоків пацієнтів. Завдяки наявності імітаційної моделі з'являється можливість не обмежувати підвищення ефективності використання ресурсів у рамках сучасної організації лікувального процесу, а також вдосконалювати саму цю організацію, моделюючи різні варіанти організаційних змін (перерозподіл ліжок між підрозділами, зміна кількості медичного персоналу, керування процесом госпіталізації і т.п.

3.2 Розробка інтерактивної середовища для роботи в реальному часі

Прикладом роботи цього типу може служити робота [3], в якій мультиагентний підхід пропонується використовувати для оперативного планування лікувального процесу у лікарні швидкої допомоги. Структура мультиагентної системи показана на Рис. 3.2.

При поступленні чергового пацієнта створюється Приймаючий агент (HA-агент), який за взаємодії з пацієнтом у реальному часі створює запис на медичній карті та передає її Агенту-ідентифікатору (IdA-агенту). Агент-ідентифікатор (IdA-агент) визначає, які спеціалісти та які ресурси потрібні в процесі лікування, а також передає інформацію Агенту моніторингу (MA-агенту) та Агенту-планувальнику (SA-агенту).

Агент-планувальник (SA-агент) складає розклад процесу лікування оптимізуючи час лікування при наявності ресурсів. Знаючи профіль необхідних спеціалістів SA-агент формує мобільну команду відповідних агентів (MSA-

агентів). У разі необхідності він переводить MSA-агентів з однієї команди в іншу. Крім того, SA-агент повідомляє MA-агенту про результати медичного втручання.

Агент моніторингу (MA-агент), попри рутинні завдання моніторингу, повідомляє IdAagentu про погіршення стану пацієнта та керує розподілом пацієнтів по палатами та лікарняним ліжкам.

Агенти мобільних медичних команд (MSA-агенти) у разі необхідності можуть за ініціативою SA-агента збиратися на консиліум, а також за його вимогою перейти до інших команд для виконання необхідних процедур. Ці агенти мають доступ до всієї інформації про пацієнтів, яка знаходиться в розпорядженні MA-агента. У розділі детально описані функції агентів різних їтипів та їх взаємодії з врахуванням реальною практики служб швидкої допомоги. Однак усі ці деталі стосуються лише програмного забезпечення, організація інтерактивної взаємодії між комп'ютером та медичним персоналом не розглядається. Кінцевою метою є доведення системи до такого рівня, щоб вона змогла здійснювати розподіл ресурсів в реальному часі.

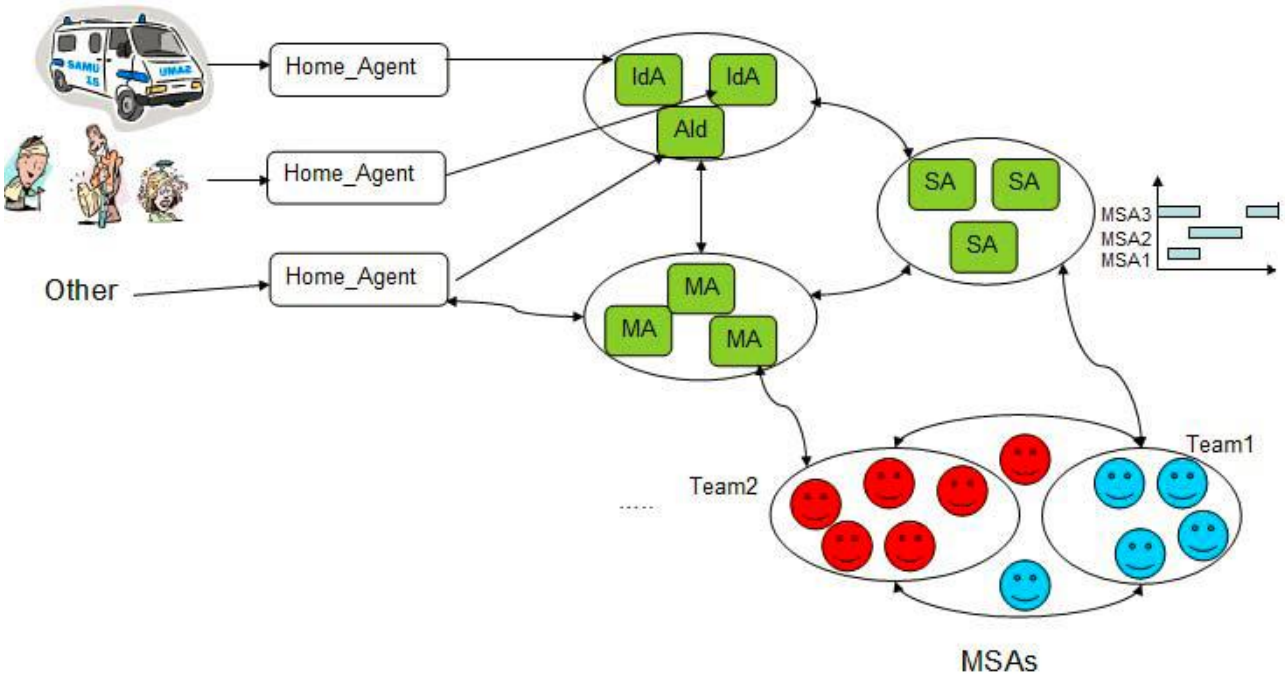

Рис. 3.2 – Мультиагентна система

3.3 Імітаційне моделювання

В [4] описана імітаційна модель відділення швидкої допомоги у складі медичного закладу (таке відділення, згідно опису, відповідає нашому травмопункту).

Агенти пацієнтів та медичногг персоналу представлені в моделі кінцевими автоматами, що мають кілька станів і переходять з одного стану в інший при поступлені на вхід сигналів від інших агентів. Наприклад, для агента-пацієнта сигналом може бути отримане від агента-лікаря призначення на рентген. Стан агента описується набором змінних, значення яких змінюються при переході від одного стану до іншого (див. Табл. 3.1).

Таблиця 3.1

Перелік змінних стану

Змінні	Можливі значення
Ім'я / ідентифікатор ID	Індивідуальне для кожного агента
Персональні дані	Вік, стать, освіта тощо.
Місцезнаходження	Реєстратура, зал очікування, кімната первинного огляду, рентгенкабінет і т.д.
Виконувані дії	Очікування, звернення за інформацією до ID, повідомлення інформації ID, перехід з одного місця в інше та ін.
Діагноз	Повний перелік наявних відхилень
Комунікабельність	Низька, середня, висока
Досвід роботи (для медперсоналу)	Малий, середній, великий

Змінна «Виконувана дія» має широкий набір значень (у таблиці показані лише три приклади), для агента кожного типу свій та залежить від місця розташування. Кожна з цих дій має визначену тривалість і вносить свій вклад у загальний час перебування пацієнта у травмпункті. Хоча в реальних системах діагноз ставиться в кінці процесу, в моделях він задається зразу, щоб можна було використати наявну статистики.

У Таблиці 3.2 представлені результати деяких досліджень на побудованій моделі: залежність результатів роботи травмпункту від кількості медичного персоналу.

Таблиця 3.2

Залежність результатів роботи від кількості медперсоналу

№	PA	TN	M	A	B	C	D	E	F	G
1	1	1	2	397	135	7.8	4.8	2.5	222	37
2	1	2	2	399	135	8.2	0.7	7.0	48	212
3	1	2	3	398	203	5.8	0.6	4.7	48	143
4	1	2	4	397	271	4.0	1.0	2.4	46	75
5	1	3	3	397	203	6.1	0.0	5.6	0	188
6	1	3	4	406	271	4.3	0.0	3.8	0	128
7	2	2	2	382	135	7.6	0.3	6.8	36	207
8	2	2	3	404	203	6.3	1.0	4.7	53	143
9	2	2	4	390	271	4.0	1.0	2.4	40	77
10	2	3	3	389	203	5.7	0.0	5.2	0	181
11	2	3	4	406	271	3.8	0.0	3.3	1	128

У таблиці введено наступні позначення. PA: к-сть адміністративного персоналу на прийомі. TN: к-сть медсестер, які виконують сортування пацієнтів. M: к-сть лікарів. A: к-сть пацієнтів, що прибули в травмпункт протягом модельованого проміжку часу. B: к-сть пацієнтів, які закінчили лікування за цей проміжок часу. Для цих пацієнтів: C: сумарний час перебування в травмпункті (в годинах), D: середній час, проведений в основному приміщенні для очікування (в годинах), E: середній час, проведений у другому приміщенні для очікування (в годинах). F: к-сть пацієнтів, які перебувають в основному приміщенні для очікування по закінченні модельованого періоду. G: к-сть пацієнтів, які перебувають в другому приміщенні для очікування по закінченні модельованого періоду.

Було проведено 11 варіантів розрахунків, в первом наборі медперсоналу – ситуація найгірша, в 11-му – найкраща. У результаті збільшення чисельності

медперсоналу на 5 осіб (додано 1 адміністратора, 2 медсестри на сортування пацієнтів та 2 лікарів) кількість обслуговуваних пацієнтів зростає на 100%, а час перебування в травмпункті зменшився на 51,3%. У варіантах 4, 6 і 9 при такій самій кількості пацієнтів час перебування зріс: на 9 хв у розрахунках 4 і 9 і на 32 хв. – у 6. Зроблено висновок, що оптимальний підбір персоналу відповідає варіанту 4, оскільки показники ефективності – найкращі, а кількість персоналу була менша на 2 особи.

У [5] описана імітаційна модель кардіохірургічного відділення медичного закладу в Ейндховені (Нідерланди). Пацієнтів цього відділення можна розділити на чотири категорії.

Пацієнти категорії I і II – планові, вони поступають в в палату загального типу, з якої їх забирають на операцію. Після операції пацієнти категорії I знаходяться в палаті терапії високого рівня, а потім, залежно від стану, повертаються до палати загального типу (у 70% випадків), або (у разі погіршення стану) переводяться на деякий час у реанімацію (15% випадків) або в палату терапії середнього рівня (15% випадків). Пацієнти категорії II після операції повертаються в палату загального типу. Пацієнти категорії III – позапланові, їм потрібна термінова операція. Пацієнти категорії IV – це пацієнти інших хірургічних відділень, яких тимчасово розміщують у кардіохірургії із-за нестачі ліжкомісць. Маршрут пацієнта категорії I показано на Рис. 3.3.

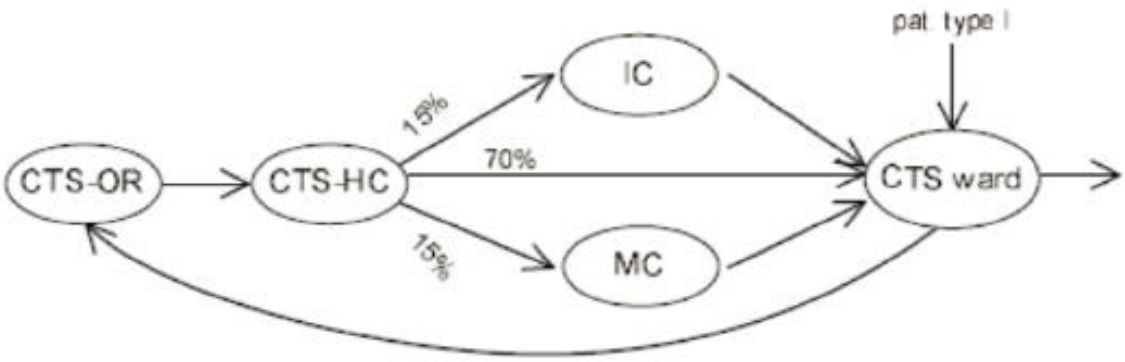

Рис. 3.3 – Маршрут пацієнта категорії I

У [5], на відміну від раніше описаної імітаційної моделі травмпункту, в отличие от ранее описаной имитационной модели травмопункта, агенти представляють не лікарів, сестер і пацієнтів, а палати різних типів. За запитами інших агентів, вони надають доступ до наявних у них ресурсів. Тривалості перебування пацієнтів у палатах різних типів описувались логнормальними розподілами, параметри яких визначаються за реальними даними методів моментів. Вхідний потік пацієнтів категорії III задався розподілом Пуассона із середнім значенням 2 пацієнта в день. Вхідний потік пацієнтів категорії IV змінюється в межах від 2 до 4 пацієнтів за день з модою 3. Вхідний потік планових пацієнтів I і II груп формується таким чином, щоб гарантувати необхідну кількість пацієнтів для операційної блоку. Про точність моделювання можна судити за наступними показниками. Пропускна здатність операційного блоку реальної системи дозволяє провести 2080 операцій за рік. Однак із-за частої відсутності вільних місць фактично виконується близько 1800 операцій. За результатами 50 прорахунків на моделях отримано 1768 операцій за рік (середнє значення) з стандартним відхиленням 40. Близькими до реальних виявилися також отримані дані про відсоток задоволених заявок щодо прийому пацієнтів категорії III і IV (82,9% і 99,0% відповідно). Точні дані про запити резервних ліжкомісць, у реальній системі відсутні. Однак дані, отримані на моделях, експерти оцінили як цілком реальні. Побудована модель використана для оцінки ефективності можливих варіантів збільшення та перерозподілу наявних в реальній системі ресурсів. На даний момент, близько 10% ще не прооперованих пацієнтів розміщують в інших відділеннях із-за відсутності місць у відділі CTS.

3.4 Висновок до розділу 3

З аналітичного огляду випливає, що найбільш цікаві для практики результати отримані в двох напрямках: у централізованому плануванні та імітаційному моделюванні за допомогою мультиагентних моделей. Однак при попередньому обговоренні з експертами, які мають досвід використання інформаційно-комп'ютерних технологій у вдосконаленні управління лікувальним

процесом щодо завдань централізованого планування практично нічого не використовується. Задачі імітаційного моделювання, очевидно, більш перспективні. Приклад імітаційного моделювання роботи кардіохірургічного відділення показує, що імітаційні моделі дійсно відкривають нові можливості для управління лікувальним процесом.

Однак у цьому напрямі зроблено лише перші кроки. Зараз імітаційні моделі розробляються великими колективами дослідників протягом тривалого часу (як правило, кілька років). Дві, описані вище, моделі – це найбільш цікаві результати з імітаційного моделювання. Специфіка багатьох медичних закладів в Україні значно ускладнює процес побудови імітаційних моделей. Тим не менше, можна сподіватися, що розробка спеціальних засобів програмування дозволить істотно підвищити швидкість та ефективність розробки таких моделей.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Забезпечення безпечних і здорових умов праці на стадії розробки агентно-орієнтованої системи для поліклінічного відділення в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

4.1 Охорона праці

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової праці. Їх праця стала більш інтенсивним, напруженим, які вимагають значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку.

В даний час комп'ютерна техніка широко застосовується у всіх областях діяльності людини. При роботі з комп'ютером на стадії розробки конфігуратора сценаріїв для моделювання і тестування режимів за допомогою оперативних та історичних технічних даних програміст піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і НВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів

робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримувати правильний режим праці та відпочинку. В іншому випадку у персоналу наголошуються значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому і хворобливі відчуття в очах, у попереку, в області шиї і руках.

Параметри мікроклімату можуть мінятися в широких межах, у той час як необхідною умовою життєдіяльності людини є підтримка постійності температури тіла завдяки терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище. Принцип нормування мікроклімату - створення оптимальних умов для теплообміну тіла людини з навколишнім середовищем.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вологості в приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 4.1) [ ].

Об'єм приміщень, в яких розміщені працівники обчислювальних центрів, не повинен бути меншим 19,5 м ^{3} / людини з урахуванням максимального числа одночасно працюючих в зміну.

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби, чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціювання повітря, опалювальна система).

Таким чином, встановлено необхідність забезпечення вимог охорони праці та техніки безпеки на робочому місці при розробці агентно-орієнтованої системи.

4.2 Безпека в надзвичайних ситуаціях

На робочому місці працініка полікланічного відділення повинні бути передбачені заходи захисту від можливого впливу небезпечних і шкідливих факторів. Рівні цих чинників не повинні перевищувати граничних значень, обумовлених правовими, технічними та санітарно-технічними нормами. Ці нормативні документи зобов'язують до створення на робочому місці умов праці, при яких вплив небезпечних і шкідливих чинників на працюючих або усунуто зовсім, або знаходиться в допустимих межах.

4.2.1 Електробезпека

Персональні комп’ютери, периферійні пристрої, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту мають відповідати класу зони, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію. Лінія електромережі для живлення персональних комп’ютерів і периферійних пристроїв виконується як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі

та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту.

У приміщенні, де одночасно експлуатуються понад п’ять персональних комп’ютерів і периферійних пристроїв, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Персональні комп’ютери і периферійні пристрої повинні підключатися до електромережі тільки за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення. У штепсельних з’єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз’єднання при відключенні має бути зворотним. Не допускається підключати персональні комп’ютери та периферійні пристрої до звичайної двопровідної електромережі, в тому числі з використанням перехідних пристроїв.

Електромережі штепсельних з’єднань та електророзеток для живлення персональних комп’ютерів та периферійних пристроїв потрібно виконувати за магістральною схемою, по 3-6 з’єднань або електророзеток в одному колі. Штепсельні з’єднання та електророзетки для напруги 12В та 42В за своєю конструкцією мають відрізнятися від штепсельних з’єднань для напруги 127В та 220В. Штепсельні з’єднання та електророзетки, розраховані на напругу 12В та 42В, мають візуально (за кольором) відрізнятися від кольору штепсельних з’єднань, розрахованих на напругу 127В та 220В. Індивідуальні та групові штепсельні з’єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах. Електромережу штепсельних розеток для живлення персональних комп’ютерів і периферійних пристроїв при розташуванні їх уздовж стін приміщення прокладають по підлозі поруч зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах, а також у пластикових коробах і пластмасових рукавах з відводами відповідно до затвердженого плану

розміщення обладнання та технічних характеристик обладнання. При розміщенні в приміщенні до п’яти персональних комп’ютерів і периферійних пристроїв допускається прокладання трипровідникового захищеного проводу або кабелю в оболонці з негорючого чи важкогорючого матеріалу по периметру приміщення без металевих труб та гнучких металевих рукавів. Не допускається в одній трубі прокладати кола до 42В та вище 42В.

При організації робочих місць операторів електромережу штепсельних розеток для живлення персональних комп’ютерів, периферійних пристроїв і у центрі приміщення прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не допускається застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, які містять сірку.

4.2.2 Запобігання виникненню надзвичайних ситуацій.

Найбільш ефективний спосіб зменшення шкоди та збитків від надзвичайних ситуацій – запобігти їх виникненню, а в разі виникнення виконувати відповідні до даної ситуації заходи.
Запобігання виникненню надзвичайних ситуацій – це підготовка та реалізація комплексу заходів, спрямованих на регулювання безпеки, проведення оцінки рівнів ризику, завчасне реагування на загрозу виникнення надзвичайної ситуації на основі даних моніторингу (спостережень), експертизи, досліджень та прогнозів щодо можливого перебігу подій з метою недопущення їх переростання у надзвичайну ситуацію або пом’якшення її можливих наслідків.

Зазначені функції запобігання надзвичайним ситуаціям техногенного і природного характеру в нашій країні покликана виконувати Єдина державна система цивільного захисту (ЄДСЦЗ), затверджена Постановою Кабінету Міністрів України від 9 січня 2014 р. №11. ЄДСЦЗ включає в себе центральні та місцеві органи виконавчої влади, виконавчі органи рад, державні підприємства, установи та організації з відповідними силами і засобами, які здійснюють нагляд за забезпеченням техногенної та природної безпеки, організують проведення роботи із запобігання надзвичайним ситуаціям і реагування у разі їх виникнення з

метою захисту населення і довкілля, зменшення матеріальних втрат. ЄДСЦЗ складається з постійно діючих функціональних та територіальних підсистем і має чотири рівні: загальнодержавний, регіональний, місцевий та об'єктовий. Кожен рівень ЄДСЦЗ має координуючі та постійні органи управління. Координуючими органами ЄДСЦЗ є:

на загально державному рівні:

- Державна комісія з питань техногенно-екологічної безпеки та надзвичайних ситуацій;

- Національна рада з питань безпечної життєдіяльності населення;

на регіональному рівні – комісії обласних державних адміністрацій з питань техногенно-екологічної безпеки та надзвичайних ситуацій;

на місцевому рівні – комісія районних державних адміністрацій і виконавчих органів рад з питань техногенно-екологічної безпеки та надзвичайних ситуацій;

на об'єктовому рівні – комісії з питань надзвичайних ситуацій об'єктів.

До систем повсякденного управління ЄДСЗР входять оснащені необхідними засобами зв'язку, оповіщення, збирання, аналізу і передачі інформації: центри управління в надзвичайних ситуаціях, оперативно-чергові служби уповноважених органів з питань надзвичайних ситуацій та цивільного захисту населення усіх рівнів; диспетчерські служби центральних та місцевих органів виконавчої влади, державних підприємств, установ та організацій.

До складу сил і засобів ЄДСЦЗ входять військові і спеціальні цивільні аварійно-рятувальні (пошуково-рятувальні) формування, які укомплектовуються з урахуванням необхідності проведення роботи в автономному режимі не менше трьох діб і перебувають у стані постійної готовності, а також недержавні (добровільні) рятувальні формування. Залежно від масштабів і особливостей надзвичайної ситуації, що прогнозується або виникла, може існувати один із таких режимів функціонування ЄДСЦЗ: повсякденної діяльності, підвищеної готовності, діяльності у надзвичайній ситуації, діяльності у надзвичайному стані. З метою ліквідації наслідків надзвичайної ситуації у мирний час може поводитися цільова мобілізація.

4.3. Висновок до розділу 4

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць має велике значення як для полегшення праці, так і для підвищення привабливості, позитивно впливаючою на продуктивність праці.

З точки зору безпеки у надзвичайних ситуаціях, то ефективність функціонування систем захисту населення і територій досягається через завчасну підготовку, оперативне реагування та ефективне управління під час надзвичайних ситуацій, своєчасне відновлення життєдіяльності населення в їх зоні.

ЗАГАЛЬНІ ВИСНОВКИ

Кваліфікаційна робота мігістра на тему «Агентно-орієнтована система для поліклінічного відділення» складається із вступу, аналітичної, основної та науково-дослідної частини, розділу «Охорона праці та безпека життєдіяльності», висновків, переліку літератури та додатків.

В аналітичній частині дано визначення програмного агента та проведений аналіз існуючих програмних аґентів. Розглянуто типи програмних аґентів. Розглянуто інструментарій для створення програмних аґентів, який включає мови і програмні засоби реалізації аґентів, мови комунікації аґентів, мови опису поведінки аґентів і законів середовища, мови представлення і управління знаннями, мови формалізації і специфікації аґентів і мультиагентних систем (МАС).

На основі аналізу відомих мов програмування програмних аґентів, встановлено, що найчастіше застосовуються мови ООП (C++, *Java*), рідше символьні і мови логічного програмування (LISP, Oz). Проаналізовано програмні платформи для створення програмних аґентів: NetLogo, StarLogo, Repast Simphony, Eclipse AMP, JADE, Jason.

В результаті цього аналізу зроблено висновок, що найбільш прийнятна платформа JADE, оскільки єдиним істотним мінусом при такому підході є необхідність детального вивчення бібліотеки. Проте використання самостійних систем не дасть нам в цьому особливої переваги, оскільки теж вимагає детального вивчення особливої мови програмування, притому досить небагатого по функціональності, що надається, і розширюваності.

Розглянуто засоби специфікацій типових моделей на прикладі спеціальної об'єктно-орієнтованої мови *RADL* (*Reticular Agent Definition Language*).

Приділено увагу мультиагентним системам та архітектурі мультиагентних застосувань. В результаті виконаної роботи можна зробити висновок, що мультиагентні системи - це один з нових перспективних напрямів штучного інтелекту.

У основній частині розглянуто питання застосування агент-орієнтованого моделювання в сфері охорони здоров'я.

У науково-дослідній частині проведено аналітичний огляд з якого випливає, що найбільш цікаві для практики результати отримані в двох напрямках: у централізованому плануванні та імітаційному моделюванні за допомогою мультиагентних моделей.

Специфіка багатьох медичних закладів в Україні значно ускладнює процес побудови імітаційних моделей. Тим не менше, можна сподіватися, що розробка спеціальних засобів програмування дозволить істотно підвищити швидкість та ефективність розробки таких моделей.

У розділі „Охорона праці та безпека в надзвичайних ситуаціях» та розглянуто питання задані консультантами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основы технологий ДО [Електронний ресурс]. – Режим доступу: http://www.websoft.ru/db/wb/42D07B203E7BFAB1C325624004EE7FF/doc.html

2. Интернет технологии в образовании - дистанционное обучение [Електронний ресурс]. – Режим доступу: http://www.curator.ru.htm

3. Multiagent Systems. A Modern Approach to Distributed Modern Approach to Artificial Intelligence. Edited by Gerhard Weiss. The MIT Press. Cambridge, Massachusetts. London, England. 1999 Massachusetts Institute of Technology.

4. Мультиагентные системы [Електронний ресурс]. – Режим доступу: http://teormin.ifmo.ru/education/intro/multiagent-systems.html

5. Самоорганизация и эволюция в открытых мультиагентных системах для холонических предприятий [Електронний ресурс]. – Режим доступу: http://eup.ru/pages/R06/Biblio/2002-08-05/F182.htm

6. Proceedings of the International Conference for Internet Technology and Secured Transactions (ICITST-2006). Editors Charles A. Shoniregun, Alex Logvynovskiy. Published by the e-Centre for Infonomics, UK.

7. Jadex Tool Guide [Електронний ресурс] / A. Pokahr, L. Braubach, R. Leppin, and A. Walczak. Distributed Systems Group. University of Hamburg, Germany, 2005.–Режим доступу: http://vsis-www.informatik.uni-hamburg.de

8. Jadex User Guide [Електронний ресурс] / A. Pokahr, L. Braubach, and A. Walczak. Distributed Systems Group. – University of Hamburg, Germany, 2005. – Режим доступу: http://vsis-www.informatik.uni-hamburg.de

9. System Architecture with XML. Berthold Daum Udo Merten. Morgan Kaufmann Publishers. San Francisco, USA, 2003 by Elsevier Science [Електронний ресурс]. – Режим доступу: http://www.mkp.com/

10. Introduction to DTD [Електронний ресурс]. – Режим доступу: http://www.w3schools.com/dtd/dtd_intro.asp

11. Модели обучения автоматизированных обучающих систем [Електронний ресурс]. – Режим доступу: http://systech.miem.edu.ru/2004/n2/Cibulskiy.htm

12. Jadex Tutorial [Електронний ресурс] / L. Braubach, A. Pokahr, and A. Walczak. Distributed Systems Group . – University of Hamburg, Germany. 2005. – Режим доступу: http://vsis-www.informatik.uni-hamburg.de

13. NIST (Nationals Institute of Standards and Technology) [Електронний ресурс]. – Режим доступу: http://www.nist.gov/

14. Проблемы реализации мультиагентных систем дистанционного обучения в сети Интернет [Електронний ресурс]. – Режим доступу: http://www.vedu.ru/info/Announce/PHT2000/thesis.asp?str=4_04

15. Использование мультиагентного онтологического подхода к созданию распределенных систем дистанционного обучения [Електронний ресурс]. – Режим доступу: http://ifets.ieee.org/russian/depository/ v7_i2/pdf/3.pdf

16. Тархов Сергей Владимирович. Система автоматизированного сетевого и дистанционного обучения с мультиагентной архитектурой [Електронний ресурс] / С.В. Тархов // Конресс конференций «Информационные технологии в образовании» . – Режим доступу: http://ito.edu.ru/2004/Moscow/III/3/III-3-4317.html

17. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – Київ: Державний комітет України по стандартизації, метрологнії та сертифікації, 1995.

18. FIPA Agent Management Specification. 1996-2002 Foundation for Intelligent Physical Agents [Електронний ресурс]. Режим доступу: http://www.fipa.org/

19. Губський А.І. Цивільна оборона / А.І. Губський. – К: Міністерство освіти, 1996. – 216 с.

ДОДАТКИ

ДОДАТОК А
Технічне завдання

Затверджую
завідувач кафедри БТ
Яворська Є.Б.

_____
“____” _____2020 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра
на тему:

Агентно-орієнтована система для поліклінічного відділення

Узгоджено:
Керівник роботи
к.т.н., доц., зав. каф. кафедри БТ
_____ Яворська Є.Б.
“__” _____ 2020 р.

Виконавець:
Студентка групи РБм-61
_____ Дороніна І.О.
“__” _____ 2020 р.

Тернопіль 2020 р.

1. НАЗВА РОБОТИ І ПІДСТАВА ДЛЯ ВИКОНАННЯ

1.1. Назва: “ Агентно-орієнтована система для поліклінічного відділення ”.

1.2. Підставою для виконання роботи є наказ по університету № 4/7-793 від «02» листопада 2020 року.

2 ВИКОНАВЕЦЬ

Студентка групи РБм-61 кафедри БТ Тернопільського національного технічного університету імені Івана Пулюя Дороніна Іванна Олексіївна.

3 МЕТА ДОСЛІДЖЕННЯ

Метою дослідження є підвищення інформатизації медичних працівників щодо перебігу процесів у медичному закладі шляхом застосування мультиагентних програмних рішень

4. ТЕХНІЧНІ ВИМОГИ

4.1. Функціональні вимоги:

4.1.1. Програмний агент системи повиннен мати змогу виконувати наступні функції:

− здатність реагування – 1) здатність агента своєчасно реагувати на зовнішнє середовище; 2) здатність агента діяти без будь-яких зовнішніх чи внутрішніх консультацій.

− автономність – означає розміщення агента в зовнішньому середовищі, причому агент є частиною цього середовища і діє в ньому постійно згідно зі своєю програмою (завданнями) так, щоб впливати в майбутньому.

− раціональність (розумність) – 1) внутрішній стан агента, коли він працює з осмисленням, передбаченням зовнішнього світу; 2) наявність у агента розумових компонентів, таких як думка, ціль, наміри, бажання.

− розважливість – 1) здатність агента до мислення; 2) використання агентом знань і мислення для визначення своїх наступних дій.

− здатність до навчання – здатність агента використовувати досвід функціонування для корегування своєї поведінки.

− комунікативність – здатність агентів співпрацювати, конкурувати або ігнорувати один одного. найчастіше має місце кооперація (або координація дій) агентів, коли агенти об’єднують свої зусилля для досягнення загальної цілі. для кооперації агентів мають бути визначені відповідні кооперaційні стратегії, колективне навчання, єрархічна організація тощо.

− тривалість функціонування – здатність агента діяти впродовж великого проміжку часу.

4.1.2. Вхідна інформація отримується шляхом:

– введення інформації користувачами.

4.1.3. Вихідна інформація:

– виводиться у вигляді глосарію

– подається у зрозумілому для користувача, лаконічному форматі.

4.2. Вимоги до надійності:

4.2.1. Сигналізація користувача в разі здійснення некоректних дій, інформування про помилку та спосіб виправлення.

4.2.2. Валідація введеної інформації.

4.3 Вимоги до апаратних засобів.

4.3.1. Система повинна працювати на будь-яких пристроях із встановленим веб-браузером.

4.3.2. Мінімальні вимоги до робочих станцій: процесор ARM Cortex-A5 або еквівалентний процесор, оперативна пам'ять від 256 Мб, об’єм пам’яті па накопичувачі: до 100 МБ; дисплей: 480 x 320; GPS і/або AGPS.

5. ЕКОНОМІЧНІ ПОКАЗНИКИ

5.1. Фінансовий сегмент включає розробку програмного агента, підтримку та адміністрування протягом терміну експлуатації, підтримку обладнання, необхідного для функціонування.

5.2. Собівартість розробки складає 5 000 грн. Вартість підтримки змінюється в залежності від термінів використання та необхідності розширення функціоналу.

6. ВИМОГИ ДО ДОКУМЕНТАЦІЇ

6.1. Для програмного продукту повинні бути розроблені наступні документи:

- Технічне завдання;
- Пояснювальна записка.

7. ПОРЯДОК ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

7.1. Стадії виконання кваліфікаційної роботи наведено в таблиці 7.1.

Таблиця 7.1 – Стадії та етапи виконання

№ етапу	Назва етапу виконання ДР	Термін виконання
1	Розроблення та затвердження технічного завдання	
2	Аналіз технічного завдання, підбір бібліографічних матеріалів, необхідних для виконання роботи	
3	Вибір протоколу передавання даних для розроблювальної мережі; вибір апаратного забезпечення	
8	Охорона праці та безпека в надзвичайних ситуаціях	
9	Оформлення	
10	Нормоконтроль	
11	Попередній захист	
12	Захист	

8. ДОДАТКОВІ УМОВИ ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

8.1. Під час виконання кваліфікаційної роботи в дане технічне завдання можна вносити зміни та доповнення.

ДОДАТОК Б

Тест план та результати тестування

Лістинг створення агента глосарію

```
GlossaryTerm.xml
<?xml version="1.0"?>
<!DOCTYPE glossary SYSTEM "GlossaryTerm.dtd">
<glossary title ="The XML- Helpbook" orderDate="2006?12?12">
<terms>
<term value="term1">
<name>dictionary</name>
<count>1</count>
<definition count="1">An abstract data type storing items, or
values. A value is accessed by an associated key. Basic operations
are new, insert, find and delete. </definition>
<giperlink>abstract data type</giperlink>
<giperlink>key</giperlink>
</term>
<term value="term2">
<name>heap</name>
<count>1</count>
<definition count="1">A complete tree where every node has a key
more extreme (greater or less) than or equal to the key of its
parent. Usually understood to be a binary heap. </definition>
<giperlink>complete tree</giperlink>
<giperlink>key</giperlink>
<giperlink>node</giperlink>
<giperlink>parent</giperlink>
<giperlink>binary heap</giperlink>
</term>
<term value="term3">
<name>linked list</name>
<count>1</count>
<definition count="1">A list implemented by each item having a link
to the next item. </definition>
<giperlink>list</giperlink>
<giperlink>link</giperlink>
</term>
<term value="term4">
<name>queue</name>
<count>1</count>
<definition count="1">A collection of items in which only the
earliest added item may be accessed. Basic operations are add (to
the tail) or enqueue and delete (from the head) or dequeue. Delete
returns the item removed. Also known as "first-in, first-out" or
FIFO. </definition>
<giperlink>head</giperlink>
<giperlink>tail</giperlink>
</term>
<term value="term5">
<name>stack</name>
<count>1</count>
```

```
<definition count="1">A collection of items in which only the most
recently added item may be removed. The latest added item is at the
top. Basic operations are push and pop. Often top and isEmpty are
available, too. Also known as "last-in, first-out" or LIFO.
</definition>
</term>
<term value="term6">
<name>tree</name>
<count>2</count>
<image ref="./tree.bmp"/>
<definition count="1">A data structure accessed beginning at the
root node. Each node is either a leaf or an internal node. An
internal node has one or more child nodes and is called the parent
of its child nodes. All children of the same node are siblings.
Contrary to a physical tree, the root is usually depicted at the top
of the structure, and the leaves are depicted at the bottom.
</definition>
<definition count="2">A connected, undirected, acyclic graph. It is
rooted and ordered unless otherwise specified. </definition>
<giperlink>node</giperlink>
<giperlink>tree</giperlink>
<giperlink>parent</giperlink>
<giperlink>root</giperlink>
<giperlink>leaf</giperlink>
<giperlink>internal node</giperlink>
<giperlink>child</giperlink>
<giperlink>siblings</giperlink>
<giperlink>connected</giperlink>
<giperlink>undirected</giperlink>
<giperlink>acyclic graph</giperlink>
<giperlink>rooted</giperlink>
<giperlink>ordered</giperlink>
</term>
<term value="term7">
<name>array</name>
<count>1</count>
<definition count="1">A set of items which are randomly accessible
by numeric index. </definition>
</term>
<term value="term8">
<name>array index</name>
<count>1</count>
<definition count="1">The location of an item in an array.
</definition>
<giperlink>array</giperlink>
</term>
<term value="term9">
<name>child</name>
<count>1</count>
<definition count="1">A node of a tree referred to by a parent node.
See the figure at tree. Every node, except the root, is the child of
some parent. </definition>
<giperlink>node</giperlink>
<giperlink>tree</giperlink>
<giperlink>parent</giperlink>
```

```
<giperlink>root</giperlink>
</term>
<term value="term10">
<name>circular list</name>
<count>1</count>
<definition count="1">A variant of a linked list in which the
nominal tail is linked to the head. The entire list may be accessed
starting at any item and following links until one comes to the
starting item again. </definition>
<giperlink>linked list</giperlink>
<giperlink>link</giperlink>
<giperlink>tail</giperlink>
</term>
<term value="term11">
<name>complete tree</name>
<count>1</count>
<definition count="1">A tree in which all leaf nodes are at some
depth n or n-1, and all leaves at depth n are toward the left.
</definition>
<giperlink>tree</giperlink>
<giperlink>depth </giperlink>
<giperlink>leaf</giperlink>
</term>
<term value="term12">
<name>connected graph</name>
<count>1</count>
<definition count="1">An undirected graph that has a path between
every pair of vertices. </definition>
</term>
<term value="term13">
<name>depth</name>
<count>1</count>
<definition count="1">Of a node, the distance from the node to the
root of the tree. </definition>
<giperlink>node</giperlink>
<giperlink>tree</giperlink>
<giperlink>root</giperlink>
</term>
<term value="term14">
<name>doubly linked list</name>
<count>1</count>
<definition count="1">A variant of a linked list in which each item
has a link to the previous item as well as the next. This allows
easily accessing list items backward as well as forward and deleting
any item in constant time. </definition>
<giperlink>linked list</giperlink>
<giperlink>link </giperlink>
</term>
<term value="term15">
<name>dynamic array</name>
<count>1</count>
<definition count="1">An array whose size may change over time.
Items are not only added or removed, but memory used changes, too.
</definition>
<giperlink>array</giperlink>
```

```
</term>
<term value="term16">
<name>height</name>
<count>1</count>
<definition count="1">The maximum distance of any leaf from the root
of a tree. If a tree has only one node (the root), the height is
zero.
</definition>
<giperlink>leaf</giperlink>
<giperlink>tree</giperlink>
<giperlink>root</giperlink>
</term>
<term value="term17">
<name>internal node</name>
<count>1</count>
<definition count="1">A node of a tree that has one or more child
nodes, equivalently, one that is not a leaf. </definition>
<giperlink>leaf</giperlink>
<giperlink>tree</giperlink>
<giperlink>node</giperlink>
<giperlink>child</giperlink>
</term>
<term value="term18">
<name>leaf</name>
<count>1</count>
<definition count="1">A node in a tree without any children. See the
figure at tree. </definition>
<giperlink>leaf</giperlink>
<giperlink>tree</giperlink>
<giperlink>node</giperlink>
<giperlink>children</giperlink>
</term>
<term value="term19">
<name>linear search</name>
<count>1</count>
<definition count="1">Search an array or list by checking items one
at a time. </definition>
<giperlink>list</giperlink>
<giperlink>array</giperlink>
</term>
<term value="term20">
<name>link</name>
<count>1</count>
<definition count="1">A reference, pointer, or access handle to
another part of the data structure. Often, a memory address.
</definition>
<giperlink>list</giperlink>
<giperlink>array</giperlink>
</term>
<term value="term21">
<name>list</name>
<count>1</count>
<definition count="1"> A collection of items accessible one after
another beginning at the head and ending at the tail. </definition>
<giperlink>head</giperlink>
```

```
<giperlink>tail</giperlink>
</term>
<term value="term22">
<name>matrix</name>
<count>1</count>
<definition count="1">A two-dimensional array. By convention, the
first index is the row, and the second index is the column.
</definition>
</term>
<term value="term23">
<name>node</name>
<count>2</count>
<definition count="1">A unit of reference in a data structure. Also
called a vertex in graphs and trees. </definition>
<definition count="2">A collection of information which must be kept
at a single memory location.
</definition>
<giperlink>graphs</giperlink>
<giperlink>trees</giperlink>
<giperlink>vertex</giperlink>
</term>
<term value="term24">
<name>order</name>
<count>4</count>
<definition count="1">The height of a tree. </definition>
<definition count="2">The number of children of the root of a
binomial tree. </definition>
<definition count="3">The maximum number of children of nodes in a
B-tree. </definition>
<definition count="4">The number of data streams, usually denoted,
in a multiway merge. </definition>
<giperlink>height</giperlink>
<giperlink>tree</giperlink>
<giperlink>children</giperlink>
<giperlink>root</giperlink>
<giperlink>node</giperlink>
</term>
<term value="term25">
<name>ordered linked list</name>
<count>1</count>
<definition count="1">A linked list whose items are kept in some
order. </definition>
<giperlink>linked list</giperlink>
</term>
<term value="term26">
<name>ordered tree</name>
<count>1</count>
<definition count="1">A tree where the children of every node are
ordered, that is, there is a first child, second child, third child,
etc.
</definition>
<giperlink>tree</giperlink>
<giperlink>children</giperlink>
<giperlink>node</giperlink>
</term>
```

```
<term value="term27">
<name>parent</name>
<count>1</count>
<definition count="1">Of a node: the tree node conceptually above or
closer to the root than the node and which has a link to the node.
See the figure at tree. </definition>
<giperlink>tree</giperlink>
<giperlink>link</giperlink>
<giperlink>root</giperlink>
<giperlink>node</giperlink>
</term>
<term value="term28">
<name>root</name>
<count>1</count>
<definition count="1">The distinguished initial or fundamental item
of a tree. The only item which has no parent. See the figure at
tree.
</definition>
<giperlink>tree</giperlink>
<giperlink>parent</giperlink>
</term>
<term value="term29">
<name>search</name>
<count>1</count>
<definition count="1">To look for a value or item in a data
structure. There are dozens of algorithms, data structures, and
approaches. </definition>
</term>
<term value="term30">
<name>self-organizing list</name>
<count>1</count>
<definition count="1">A list that reorders the elements based on
some self-organizing heuristic to improve average access time.
</definition>
<giperlink>list</giperlink>
<giperlink>self-organizing heuristic </giperlink>
</term>
<term value="term31">
<name>sibling</name>
<count>1</count>
<definition count="1">A node in a tree that has the same parent as
another node is its sibling. </definition>
<giperlink>tree</giperlink>
<giperlink>parent</giperlink>
<giperlink>node</giperlink>
</term>
<term value="term32">
<name>sorted array</name>
<count>1</count>
<definition count="1">An array whose items are kept sorted, often so
searching is faster. </definition>
<giperlink>array</giperlink>
</term>
<term value="term33">
<name>sorted list</name>
```

```
<count>1</count>
<definition count="1">A list whose items are kept sorted.
</definition>
<giperlink>list</giperlink>
</term>
<term value="term34">
<name>square matrix</name>
<count>1</count>
<definition count="1">A n*n matrix, i.e., one whose size is the same
in both dimensions. </definition>
<giperlink>matrix</giperlink>
</term>
<term value="term35">
<name>tail</name>
<count>2</count>
<definition count="1">The last item of a list. </definition>
<definition count="2">All but the first item of a list; the list
following the head. </definition>
<giperlink>list</giperlink>
</term>
</terms>
</glossary>
```

Лістинг опису тегів XML документу в форматі DTD

```
GlossaryTerm.dtd
<?xml version="1.0" encoding="windows-1251"?>
<!ELEMENT glossary (terms)>
<!ATTLIST glossary
title CDATA #REQUIRED
orderDate CDATA "">
<!ELEMENT terms (term+)>
<!ELEMENT term (name,definition+)>
<!ATTLIST term
value CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT count (#PCDATA)>
<!ELEMENT image EMPTY>
<!ATTLIST image
ref CDATA #REQUIRDE>
<!ELEMENT definition (#PCDATA)>
<!ATTLIST definition
count CDATA #REQUIRDE>
<!ELEMENT giperlink (#PCDATA)>
```

ДОДАТОК В

Апробація результатів досліджень

III Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 654.16 : 616.12-073.7
Гринчук К., Дороніна І. – ст. гр. $РБ_м$–51
Тернопільський національний технічний університет імені Івана Пулюя

СПЕЦИФІКАЦІЯ МЕТОДІВ ПЕРЕДАЧІ БІОМЕДИЧНИХ СИГНАЛІВ

Науковий керівник: к.т.н., доц. Є.Б. Яворська

Hrynchuk K., Doronina I.
Ternopil Ivan Puluj National Technical University

SPECIFICATION OF METHODS OF TRANSMISSION OF BIOMEDICAL SIGNALS

Supervisor: assoc. prof. E. Yavorska

Ключові слова: біотехнічна система, біооб'єкт, передача сигналу.
Keywords: biotechnical system, bio-object, signal transmission.

У сучасних медичних дослідженнях не можливо обійтись без використання біотехнічних апаратно-програмних засобів відбору, опрацювання, зберігання медико-біологічної інформації. Століттями медики для діагнозу та досліджень відбирали інформацію, в основному, за допомогою своїх п'яти почуттів. На даний час з цією метою використовують вимірні перетворювачі та електроди, давачі, засоби передачі та зв'язку, процесори для обробки сигналів, запам'ятовуючі пристрої тощо. В основі роботи біомедичної апаратури лежить використання біомедичних сигналів – зміни у просторі і часі фізичних величин, властивих об'єктові, якщо ці зміни є інформативними (містять дані, за якими можна скласти уяву про стан або впливати на стан біооб'єкту). Сукупність технічних засобів і тракту для передачі повідомлення на відстані називають каналом зв'язку. Передача по заданому каналу відбувається незалежно від інших каналів. Канали зв'язку організовуються у лінії зв'язку . Сукупність ліній зв'язку, які працюють на спільній для багатьох абонентів частоті або групі частот утворюють мережу. Головною відмінністю біотехнічної системи від усіх інших систем, є те, що джерелом сигналів є біооб'єкт. Давач є посередником з живим об'єктом, він повинен якомога менше впливати на сигнал. Звичайно давач неможливо прямо під'єднати до пристрою візуалізації. Сигнал з його виходу необхідно підсилити, за допомогою аналогово-цифрового перетворювача перетворити в двійковий код і через порт обміну передати в мобільний телефон. З мобільного телефона сигнал через мережу мобільного зв'язку стандарту GSM 900/1800 поступає на систему яка складається з мобільного телефона і через USB порт під'єднана до PC. З допомогою персонального комп'ютера відбувається обробка сигналу і порівняння його характеристик з етлоном чи якимось іншим методом судять про стан сигналу і інші характеристики біооб'єкту, здійснюючи діагностику.

Одна з основних тенденцій сучасної медицини – широке впровадження техніки, не тільки закономірна, але й необхідна, оскільки за допомогою сучасних технічних засобів лікар став інтелектуально потужнішим, отримує нові засоби лікувального впливу, підсилює свої сенсорні властивості, все ближче підходить до оптимального керування процесами, які відбуваються в організмі людини під час лікування.