Co-funded by the
Erasmus+ Programme
of the European Union

GAMEHUB University Enterprises Cooperation
In Game Industry In Ukraine

ПРОЕКТ ЕРАЗМУС+
GAMEHUB: «СПІВРОБІТНИЦТВО МІЖ УНІВЕРСИТЕТАМИ
ТА ПІДПРИЄМСТВАМИ В СФЕРІ ГРАЛЬНОЇ ІНДУСТРІЇ В УКРАЇНІ»

ERASMUS+ PROJECT
GAMEHUB: «UNIVERSITY-ENTERPRISES COOPERATION
IN GAME INDUSTRY IN UKRAINE»

МУЛЬТИАГЕНТНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ В ІГРОВИХ ДОДАТКАХ

Co-funded by the
Erasmus+ Programme
of the European Union

GAMEHUB University Enterprises Cooperation In Game Industry In Ukraine

GAMEHUB: «СПІВРОБІТНИЦТВО МІЖ УНІВЕРСИТЕТАМИ
ТА ПІДПРИЄМСТВАМИ В СФЕРІ ГРАЛЬНОЇ ІНДУСТРІЇ В УКРАЇНІ»

МУЛЬТИАГЕНТНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ В ІГРОВИХ ДОДАТКАХ

Харків
«Друкарня Мадрид»
2018

УДК 004.9
ББК 32.973-018.2
в6
К 28

Проект ЕРАЗМУС+
GameHub: «Співробітництво між університетами та підприємствами
в сфері гральної індустрії в Україні»
№ 561728-EPP-1-2015-1- ES-EPPKA2-CBHE-JP

Касілов О.В.

К 28 Мультиагентні системи та технології в ігрових додатках: довідник модуля. / О.В.Касілов. - Х.: «Друкарня Мадрид», 2018. - 82 с.
ISBN 978-617-7683-03-1

Довідник модуля «Мультиагентні системи та технології в ігрових додатках» написаний згідно з останніми освітньо — кваліфікаційними вимогами до підготовки магістрів зі спеціальності «Інформаційні технології». З урахуванням кредитно-модульної системи навчальна дисципліна формується як система змістовних модулів. Довідник модуля складається з інформації стосовно модуля та критеріїв оцінювання, лекційних та лабораторних робіт.
Призначено для студентів всіх спеціальностей вузів, викладачів курсу «Мультиагентні системи та технології в ігрових додатках», фахівців у галузі «Інформаційні технології», які підвищують кваліфікацію.

Бібліографічні описи здійснено відповідно до існуючих державних та міждержавних стандартів: ДСТУ ГОСТ 7.1:2006 «Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання».

The module manual "Multiagent systems and technologies in gaming applications" is written in accordance with the latest educational and qualification requirements for the preparation of masters in the "Information Technologies" specialty. Taking into account the credit-module system, this discipline is formed as a system of content modules. The module manual consists of information on the module and criteria for evaluation, lecture and laboratory work.
It is intended for students of all specialties of higher educational institutions, lecturers of the "Multiagent systems and technologies in gaming applications" course, specialists in the field of "Information Technologies", who enhance their

This publication has been funded with support from the European Union. The publication reflects the views only of the authors, and the Union cannot be held responsible for any use which may be made of the information contained therein.

УДК 004.9
ББК 32.973-018.2

ISBN 978-617-7683-03-1

ЗМІСТ

ДОВІДНИК МОДУЛЯ

ЛЕКЦІЇ

ЛАБОРАТОРНІ РОБОТИ

Довідник модуля

1 ВСТУП

Предметом навчальної дисципліни є вивчення теоретичних основ застосування агентних технологій та мультиагентних систем; оволодіння методологією побудови моделей поведінки програмних агентів; придбання практичних навичок створення програмних агентів; вивчення мов проектування програмних агентів та програмного інструментарію для створення мультиагентного простору.

МЕТА ДИСЦИПЛІНИ

Метою викладання навчальної дисципліни «Мультиагентні системи і технології» є формування у студентів цілісної системи знань про агентні технологій та мультиагентні системи, навчання студентів методам і засобам проектування, створення і роботи з програмними агентами та мультиагентними системами.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

Основними завданнями вивчення дисципліни «Мультиагентні системи і технології» є набуття знань про сучасну теорію побудови мультиагентних систем та програмних агентів, методи та технології їх розробки, а саме: концепцію та технологію мультиагентних систем; логічний рівень опису програмних агентів; моделі, класи, моделі поведінки програмних агентів та вмінь проектувати інформаційні системи із застосуванням програмних агентів, реалізовувати технологію програмних агентів сучасними програмними засобами; використовувати технологію програмних агентів для роботи в інформаційних системах підтримки прийняття рішень.

Змістовний модуль «Мультиагентні системи та технології в ігрових додатках» дисципліни «Мультиагентні системи і технології» орієнтовно на оволодіння студентами практичних навичок використання середовища розробки додатків для мобільних пристроїв та застосування Android SDK, Eclips IDE та JDK.

2 ОПИС МОДУЛЯ

Галузь знань: 12 "Інформаційні технології".

Спеціальність: 122 "Комп'ютерні науки та інформаційні технології".

Рівень: магістр.

Назва дисципліни: Інтелектуальні системи підтримки прийняття рішень.

Назва змістовного модуля: Мультиагентні системи та технології в ігрових додатках.

Семестр: 9

Кількість кредитних одиниць: дисципліна – 4,0, модуль – 1,0

Орієнтовна кількість годин: дисципліна – 120, модуль – 30

Викладачі: доцент, к.т.н. Касілов О.В.; ст. викладач, к.т.н. Заволодько А.Є.

3 ПЕРЕЛІК КОМПЕТЕНТНОСТЕЙ ТА РЕЗУЛЬТАТИ НАВЧАННЯ[1]

ЗАГАЛЬНІ (УНІВЕРСАЛЬНІ) КОМПЕТЕНТНОСТІ

ЗК-1	Здатність розв'язувати складні задачі і проблеми у певній галузі професійної діяльності або у процесі навчання, що передбачає проведення досліджень та/або здійснення інновацій та характеризується невизначеністю умов і вимог.
ЗК-2	Здатність володіння спеціалізованими концептуальними знаннями на рівні новітніх досягнень, які є основою для оригінального мислення та інноваційної діяльності, зокрема в контексті дослідницької роботи.
ЗК-4	Здатність до розв'язання складних задач і проблем, що потребує оновлення та інтеграції знань, часто в умовах неповної/ недостатньої інформації та суперечливих вимог.

СПЕЦІАЛЬНІ (ФАХОВІ) КОМПЕТЕНТНОСТІ

ПК-5	Здатність до системного мислення, застосування методології системного аналізу для дослідження складних проблем різної природи, методів формалізації та розв'язанні системних задач, що мають суперечливі цілі, невизначеності та ризики.
ПК-7	Мати розуміння сучасних можливостей та технологій комп'ютерної графіки, існуючих програмних засобів для моделювання спеціальних ефектів та побудови геометричних образів.
ПК-10	Мати розуміння математичних основ формулювання задач пошуку оптимальних рішень в рамках розв'язання проблем проектування програмного забезпечення. Здатність володіти методами, алгоритмами та їхнею програмною реалізацію, спрямованою на розв'язання задачі пошуку мінімуми функції мети.

ПРОГРАМНІ РЕЗУЛЬТАТИ НАВЧАННЯ

ЗАГАЛЬНА ПІДГОТОВКА

РНз-1	Знання методів розв'язання складних задач і проблем у певній галузі професійної діяльності або у процесі навчання, що передбачає проведення досліджень та/або здійснення інновацій та характеризується невизначеністю умов і вимог.
РНз-2	Володіння спеціалізованими концептуальними знаннями на рівні новітніх досягнень, які є основою для оригінального мислення та інноваційної діяльності, зокрема в контексті дослідницької роботи.
РНз-4	Знання методів розв'язання складних задач і проблем, що потребує оновлення та інтеграції знань, часто в умовах неповної/ недостатньої інформації та суперечливих вимог.

ПРОФЕСІЙНА ПІДГОТОВКА

РН-5	Володіння системним мисленням, застосування методології системного аналізу для дослідження складних проблем різної природи, методів формалізації та розв'язанні системних задач, що мають суперечливі цілі, невизначеності та ризики.
РН-7	Знати та розуміти сучасні можливості та технології комп'ютерної графіки, існуючих програмних засобів для моделювання спеціальних ефектів та побудови геометричних образів.
РН-10	Знати та розуміти математичні основ формулювання задач пошуку оптимальних рішень в рамках розв'язання проблем проектування програмного забезпечення. Володіти методами, алгоритмами та їхнею програмною реалізацію, спрямованою на розв'язання задачі пошуку мінімуми функції мети.

[1] Профіль програми освітнього ступенів бакалавра. Спеціальність 122 Комп'ютерні науки та інформаційні технології, спеціалізація 122-09 Системи штучного інтелекту

РНс9-2	Вміння організовувати самостійну, індивідуальну роботу, здійснювати комплексні дослідження та прийняття рішень в міждисциплінарних областях.

ПЕРЕЛІК КОМПЕТЕНТНОСТЕЙ ТА РЕЗУЛЬТАТИ НАВЧАННЯ МОДУЛЬ «МУЛЬТИАГЕНТНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ В ІГРОВИХ ДОДАТКАХ»

Загальні (універсальні) компетентності	ЗК-1, ЗК-2, ЗК-4
Спеціальні (фахові) компетентності	ПК-5, ПК-7, ПК-10
Загальна підготовка	РНз-1, РНз-2, РНз-4
Професійна підготовка	РН-5, РН-7, РН-10, РНс9-2

4 МІЖДИСЦИПЛІНАРНІ ЗВ'ЯЗКИ

Для засвоєння матеріалу використовується такий перелік забезпечуючих дисциплін:
- Дискретна математика.
- Основи баз даних та знань (Моделі даних та знань, системи підтримки прийняття рішень, видобування даних, інтелектуальний аналіз даних).
- Основи програмування та алгоритмічні мови.
- Об'єктно-орієнтоване програмування.

5 МЕТА ТА ПЕРЕДБАЧУВАНІ РЕЗУЛЬТАТИ ВИВЧАННЯ МОДУЛЯ

5.1 МЕТА МОДУЛЯ

Мета модуля – формування у студентів системи знань про агентні технології та мультиагентні системи, навчання студентів методам і засобам проектування, створення і робота з програмними агентами та мультиагентними системами та застосування їх в мобільних додатках (іграх).

5.2 РЕЗУЛЬТАТИ НАВЧАННЯ

ЗНАННЯ ТА ЇХ ВИКОРИСТАННЯ

У разі успішного оволодіння матеріалами модуля, студент буде вміти використовувати сучасну теорію побудови мультиагентних систем та програмних агентів, методи та технології їх розробки, а саме: концепцію та технологію мультиагентних систем; логічний рівень опису програмних агентів; моделі, класи, моделі поведінки програмних агентів та вмінь проектувати інформаційні системи із застосуванням програмних агентів, реалізовувати технологію програмних агентів сучасними програмними засобами; використовувати технологію програмних агентів для роботи в інформаційних системах підтримки прийняття рішень.

ДОСЛІДНИЦЬКІ НАВИЧКИ

У разі успішного вивчення модуля, студент буде вміти аналізувати можливості програмних додатків для створення програмних агентів та проектувати інформаційні системи із застосуванням програмних агентів, реалізовувати технологію програмних агентів сучасними програмними засобами; використовувати технологію програмних агентів для роботи в інформаційних системах підтримки прийняття рішень.

СПЕЦІАЛЬНІ ВМІННЯ

У разі успішного вивчення модуля, студент буде вміти:

- визначати необхідні інструментальні засоби для створення програмних агентів;
- використовувати можливості програмного середовища розробки додатків Eclips IDE, Android SDK та JDK для створення програмних агентів, додатків та ігр;
- приймати участь у розробці комп'ютерних програмних агентів, ігрових додатків та інформаційних систем підтримки прийняття рішень.

СОЦІАЛЬНІ ВМІННЯ

У разі успішного вивчення модуля студент буде вміти приймати участь в командній роботі: обговорювати в групі шляхи побудови комп'ютерних програмних агентів та ігрових додатків.

ОСОБИСТІ ЯКОСТІ

У разі успішного вивчення модуля студент буде вміти:

- самостійно вирішувати питання, які відносяться до побудови та створення програмних агентів, їх застосування у комп'ютерних ігрових додатках;
- аналізувати сучасну науково-технічну літературу з питань створення програмних агентів та їх застосування, в тому числі у комп'ютерних ігрових додатках.

6 КАЛЕНДАРНИЙ ПЛАН СЕМЕСТРУ І СТРУКТУРА МОДУЛЯ

ІНФОРМАЦІЙНЕ НАПОВНЕННЯ ЗМІСТОВНОГО МОДУЛЯ

Номер	Номер тижня	Тема та зміст модуля
1	11-12	Мультиагентні системи і технології. Мультиагентний підхід. Мультиагентні системи. Агенти. Установка ПЗ для розробки і тестування додатків.
2	13-14	Розробка мультиагентних систем. Мультиагентні системи. Сучасні міжнародні стандарти створення агентів і платформи МАС. Платформи для розробки МАС. Області застосування. Розробка програми.
3	15-16	Проект «Мультиагентна система». Мультиагентний підхід і трьохрівнева система управління. Агент для автономної групи. Алгоритми групової роботи мережі. Програмування мікрокомп'ютерів. Ознайомлення з JADE. Приклад мультиагентного додатки з JADE-агентом на платформі Android.

7 ФОРМИ НАВЧАННЯ

Навчальне навантаження модуля складається з аудиторної та самостійної роботи. Аудиторна робота включає 3 лекції та 4 лабораторні роботи. Самостійна робота студентів передбачає підготовку до аудиторних занять і лабораторних робіт, а також підготовку до тесту та оформлення залікового звіту з лабораторних робіт.

Підготовка до поточних аудиторних занять є аналіз літератури, інтернет-матеріалів по темам лекцій і лабораторних робіт, підготовка до тестів.

Контактні години передбачають індивідуальні консультації та контроль студентів в онлайн режимі.

Заліковий звіт – опис та презентація за результатами виконання лабораторних робіт 1 – 4. – Мультиагентний додаток на платформі Android.

8 ПОРЯДОК ПРОВЕДЕННЯ АТЕСТАЦІЇ

Загальний принцип оцінювання підсумкових знань студента з курсу "Мультиагентні системи і технології" полягає в тестуванні студентів на лекціях, оцінці поточної практичної роботи студента у навчальному семестрі на лабораторних роботах та оцінки контрольного заходу у формі іспиту, у результаті яких студент має сумарну оцінку в балах.

За результатами вивчення кожного змістовного модуля передбачено оцінка виконання залікового завдання. Оцінка включає: результати тестування студентів на лекціях, результати поточного опитування при виконанні лабораторних робіт модуля, оцінку за виконання залікового завдання. Сумарно оцінка кожного змістовного модуля не може бути більш ніж 30 балів. Максимальна оцінка іспиту 40 балів.

ОЦІНКА РЕЗУЛЬТАТІВ ВИВЧЕННЯ ДИСЦИПЛІНИ В ЦІЛОМУ

Поточне тестування	Балів
Змістовний модуль 1	до 30
Змістовний модуль 2	до 30
Іспит	до 40
Разом	До 100

ГРАФІК ПРОВЕДЕННЯ ПОТОЧНОГО ОЦІНЮВАННЯ МОДУЛЯ

Номер тижня	Оцінювання
13	Оцінка виконання лабораторної роботи 1
14	Оцінка виконання лабораторної роботи 2
15	Оцінка виконання лабораторної роботи 3
16	Оцінка виконання лабораторної роботи 4 Оцінка залікового звіту

ПРЕДСТАВЛЕННЯ ЗВІТУ ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

При представленні звіту з лабораторних робіт необхідно виконувати наступне. При умові виконання кожної окремої лабораторної роботи єдиний звіт надається студентом на 16 тижні семестру. Продовження терміну можливе лише при наявності поважної причини, передбаченої порядком навчання студентів у вищій школі. При цьому:

електронна версія єдиного звіту за результатами проходження модуля надається викладачеві через систему Інтернет на 9 та 16 тижні;

під час виконання 4-ї лабораторної роботи на 4 тижні студент демонструє закінчений варіант мультиагентної системи;

за кожний день прострочки представлення та здачі єдиного звіту по модулю знімається 1 бал (не більш ніж 5 днів – 5 балів).

МЕТОД ОЦІНКИ ЗМІСТОВНОГО МОДУЛЯ

Кількість балів в загальній оцінці змістовного модулю відповідає наступному:

Тестування по модулю	максимально 2 бал
Виконання лабораторної роботи 1	максимально 6 бали
Виконання лабораторної роботи 2	максимально 6 бали
Виконання лабораторної роботи 3	максимально 6 бали
Виконання лабораторної роботи 4	максимально 6 бали
Оцінка залікового звіту	максимально 4 бали

Усі набрані бали підсумовуються (максимально 30 балів), штрафні бали за запізнення в представленні єдиного звіту (максимально мінус 5 балів) віднімаються. Сумарна оцінка (від 0 до 30 балів) є індивідуальна оцінка студента освоєння змістовного модуля.

МЕТОД ОЦІНКИ ДИСЦИПЛІНИ В ЦІЛОМУ

Оцінки студентів за результатами вивчення змістовних модулів 1 – 2 підсумовуються. Оцінка іспиту (максимально 40 балів) додається. Таким чином розраховується сумарна оцінка студента в балах за дисципліною.

Сумарна оцінка в балах переводиться за нижченаведеною шкалою оцінювання в національну та ЄКТС- оцінку:

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою
90 – 100	A	Відмінно
82 – 89	B	Дуже добре
74 – 81	C	Добре
64 – 73	D	Задовільно
60 – 63	E	Посередньо
35 – 59	FX	Не зараховано з можливістю повторного складання
0 – 34	F	Не зараховано з обов'язковим повторним вивченням дисципліни

КРИТЕРІЇ ОЦІНЮВАННЯ ПРЕЗЕНТАЦІЇ РЕЗУЛЬТАТІВ РОБОТИ

	Високий	Середній		Низький
	4	3	2	1
організація	Організаційна структура (конкретне введення і висновок, послідовність викладу матеріалу всередині тіла і переходи) чітко і послідовно спостерігалася, вміло представлена і робить зміст презентації цілісним.	Організаційна структура (конкретне введення і висновок, послідовність викладу матеріалу всередині тіла і переходи) чітко і послідовно проглядається в презентації.	Організаційна структура (специфічне введення і висновок, послідовність викладу матеріалу всередині тіла і переходи) періодично проглядається в презентації.	Організаційна структура (специфічне введення і висновок, послідовність викладу матеріалу всередині тіла і переходи) не спостерігається в презентації.
Мова	Вибір мови є вражаючим, що запам'ятовується, є привабливим і підвищує ефективність презентації. Мова в презентації підходить аудиторії.	Вибір мови є продуманим і в цілому підтримує ефективність презентації. Мова в презентації підходить аудиторії.	Вибір мови є повсякденним і звичайним і частково підтримує ефектність презентації. Мова в презентації підходить аудиторії.	Вибір мови незрозумілий і мінімально підтримує ефективність презентації. Мова в презентації не підходить аудиторії.
Подача	Вміння подачі (поза, жест, зоровий контакт і голосова виразність) роблять презентацію переконливою, а доповідач виглядає підготовленим, впевненим.	Вміння подачі (поза, жест, зоровий контакт і голосова виразність) роблять презентацію цікавою, і динаміка доповідача виглядає впевнено.	Вміння подачі (поза, жест, зоровий контакт і голосова виразність) роблять презентацію зрозумілою, а доповідач виглядає не дуже впевнено.	Вміння подачі (поза, жест, зоровий контакт і голосова виразність) роблять презентацію незрозумілою, а доповідь виглядає невпевнено.
Допоміжні матеріали	Різноманітні типи допоміжних матеріалів (пояснення, приклади, ілюстрації, статистика, аналоги, цитати з відповідних джерел) дають посилання на інформацію або аналіз, які в значній мірі підтримують презентацію і підтверджують авторитет доповідача в представленій темі.	Допоміжні матеріали (пояснення, приклади, ілюстрації, статистика, аналогії, цитати з відповідних джерел) містять відповідні посилання на інформацію або аналіз, які в цілому підтримують презентацію і підтверджують авторитет доповідача в представленій темі.	Допоміжні матеріали (пояснення, приклади, ілюстрації, статистичні дані, аналоги, цитати з відповідних джерел) містять відповідні посилання на інформацію або аналіз, які частково підтримують презентацію і авторитет доповідача в представленій темі.	Недостатні допоміжні матеріали (пояснення, приклади, ілюстрації, статистика, аналоги, цитати з відповідних джерел) містять відповідні посилання на інформацію або аналіз, які мінімально підтримують презентацію і авторитет доповідача в представленій темі.
Головна думка	Головна думка є переконливою (точно сформульована, відповідним чином повторюється, запам'ятовується і міцно підтримується).	Головна думка є ясною і узгоджується з допоміжним матеріалом.	Головна думка є в принципі зрозумілою, але не часто повторюється і не запам'ятовується.	Головна думка може бути виведена, але явно не вказана в презентації.

9 ЗВОРОТНИЙ ЗВ'ЯЗОК

Інформація щодо результатів тестування, виконання лабораторних робіт та загальна оцінка змістовного модулю надається кожному студенту як індивідуально, так і групі в цілому.

Інформація щодо результатів тестування надається студентам по завершенню тестування. Загальні результати надаються на 16 тижні навчання.

Інформація щодо оцінки виконання лабораторної роботи надається студентові під час заняття.

Інформація щодо оцінки змістовного модуля в цілому надається студентам на 9 та 16 тижні навчання.

Контактні дані для on-line допомоги та консультування:

Викладачі: доцент, к.т.н. Касілов О.В., e-mail: o.kasilov@gmail.com

старший викладач, к.т.н. Заволодько Г.Е., e-mail: ann.zavolodko@gmail.com

10 ВИКЛАДАЦЬКИЙ СКЛАД ТА ДОПОМІЖНІ ДЖЕРЕЛА

ОБОВ'ЯЗКИ ВИКЛАДАЧІВ

- подача матеріалів модуля згідно з програмою;
- оцінка результатів тестування та виконання лабораторних робіт.

ОБОВ'ЯЗКИ КООРДИНАТОРА ДИСЦИПЛІНИ

- планування та внесення змін до модуля;
- координація і управління професорсько-викладацьким складом;
- координація проведення тестування, лабораторних робіт та іспиту.

ОБОВ'ЯЗКИ ДОПОМІЖНОГО ПЕРСОНАЛУ

Допоміжний персонал здійснює підготовку комп'ютерної техніки до виконання лабораторних робіт студентами та надає технічну підтримку під час виконання лабораторних робіт.

КОНТАКТНІ ДАНІ ВИКЛАДАЧІВ

доцент, к.т.н. Касілов О.В., e-mail: o.kasilov@gmail.com

старший викладач, к.т.н. Заволодько Г.Е., e-mail: ann.zavolodko@gmail.com

КОНТАКТНІ ДАНІ КУРАТОРА

доцент, к.т.н. Касілов О.В., e-mail: o.kasilov@gmail.com

11 НАВЧАЛЬНА ПРОГРАМА І МАТЕРІАЛИ

11.1 ТЕМА 1: МУЛЬТИАГЕНТНІ СИСТЕМИ І ТЕХНОЛОГІЇ

АНОТАЦІЯ

Лекція знайомить з основними поняттями «Агент», мультиагентна система, мультиагентне керування, мультиагентний підхід, емерджeнальний інтелект, інтелектуальний агент та його властивості.

МЕТА ЛЕКЦІЇ

Ознайомити студентів з поняттями «Агент», мультиагентна система, мультиагентне керування, мультиагентний підхід, емерджeнальний інтелект, інтелектуальний агент, властивості інтелектуального агента, сфери застосування інтелектуальних агентів, побудова складних систем з застосуванням мультиагентів.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

Студент оволодіє поняттями: агент, мультиагентна система, мультиагентне керування, мультиагентний підхід, емерджeнальний інтелект, інтелектуальний агент. Буде знати основні властивості інтелектуального агента та методи їх досягнення агентом, сфери застосування інтелектуальних агентів. Зможе побудувати елементи складної системи з застосуванням мультиагентів.

КОНТРОЛЬНІ ЗАПИТАННЯ

Дайте визначення «мультиагентна система».

Що лежить в основі мультиагентного підходу?

У чому полягає суть мультиагентних технологій?

Що таке емерджeнальний інтелект, наведіть приклади?

У чому відмінності мультиагентної та традиційної системи побудови програмного забезпечення?

Дайте деякі з визначень «Агент».

Що представляє собою інтелектуальний агент?

Якими властивостями повинен володіти інтелектуальний агент?

Дайте визначення властивості «реактивність» інтелектуального агента та методи досягнення цих властивостей агентом.

Дайте визначення властивості «проактивність» інтелектуального агента та методи досягнення цих властивостей агентом.

Дайте визначення властивості «соціальність» інтелектуального агента та методи досягнення цих властивостей агентом.

Дайте визначення властивості «адаптивність» інтелектуального агента.

Побудова складних систем з застосуванням мультиагентнів.

Сфери застосування інтелектуальних агентів.

МЕТОДИЧНІ МАТЕРІАЛИ ТА ВКАЗІВКИ

http://blogs.kpi.kharkov.ua/v2/gamehub/download

11.2 ЛАБОРАТОРНА РОБОТА № 1. УСТАНОВКА ПЗ ДЛЯ РОЗРОБКИ ТА ТЕСТУВАННЯ ДОДАТКІВ

АНОТАЦІЯ

Лабораторна робота орієнтована на ознайомлення студентів з основними елементами середовища розробки додатків для ОС Android. Середовище розробки Eclips IDE, Android SDK та JDK.

МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Освоїти основні прийоми розробки додатків для ОС Android, вивчити процес встановлення і налаштування середовища розробки, знайомство з Android Virtual Device та призначеним для користувача інтерфейсом (UI), проведення сеансу налагодження.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

У разі успішного виконання лабораторної роботи, студент буде вміти використовувати середовища розробки додатків Eclips IDE, Android SDK та JDK для ОС Android, зберігати та завантажувати проект розробки, буде знати та вміти застосовувати основні прийоми роботи з інструментами що входять до складу Andrid SDK.

КОНТРОЛЬНІ ЗАПИТАННЯ

Засоби та платформи розробки мобільних додатків, основні функціональні можливості.

Призначення віртуального пристрою.

Перерахуйте основні елементи інтерфейсу що застосовуються в мобільних додатках.

Основні компоненти середовища відладки Android додатка.

Призначення XML кода при створенні мобільного додатка.

Життєвий цикл додатка.

МЕТОДИЧНІ МАТЕРІАЛИ ТА ВКАЗІВКИ

http://blogs.kpi.kharkov.ua/v2/gamehub/download

11.3 ТЕМА 2. РОЗРОБКА МУЛЬТИАГЕНТНИХ СИСТЕМ

АНОТАЦІЯ

Лекція знайомить з основами архітектури та побудови мультиагентних систем, формами взаємодії між агентами, сучасними міжнародними стандартами створення агентів та МАС, платформами для розробки МАС та областями застосування МАС.

МЕТА ЛЕКЦІЇ

Ознайомити студентів з основами архітектури та побудови мультиагентних систем, формами взаємодії між агентами, сучасними міжнародними стандартами створення агентів та платформи для їх розробки, областями застосування МАС.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

Сформувати у студентів знання щодо основних принципів побудови мультиагентних систем, форм взаємодії між агентами. Навчити використовувати сучасні міжнародні стандарти створення агентів, платформи для розробки МАС та області застосування МАС.

КОНТРОЛЬНІ ЗАПИТАННЯ

Мультиагентні системи та їх компоненти.

Взаємодії між агентами.

Структура типового агента.

Архітектура ядра мультиагентної системи.

Методологія висхідного еволюційного проектування МАС.

Методологія спадного проектування МАС.

Стандарти створення агентів і платформи МАС.

Стандарт OMG MASIF.

Специфікації FIPA.

Розробки DARPA в області розподілу знань.

Мови комунікації агентів.

Агентні платформи.

Система управління агентами.

JADE середовище розробки МАС.

JACK платформа створення МАС.

Madkit засіб створення МАС.

Області застосування МАС.

Надати приклади МАС, які використовуються в пошукових системах.

Перерахуйте основні функції МАС в задачах автоматизації управління ресурсами підприємства.

Схема взаємодії модулів інструментальної системи.

МЕТОДИЧНІ МАТЕРІАЛИ ТА ВКАЗІВКИ

http://blogs.kpi.kharkov.ua/v2/gamehub/download

11.4 ЛАБОРАТОРНА РОБОТА № 2. РОЗРОБКА МОБІЛЬНОГО ДОДАТКА

АНОТАЦІЯ

Лабораторна робота орієнтована на оволодіння студентами навичок розробки мобільних додатків.

МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Розробка мобільного додатка що дозволяє отримувати фотографії з вбудованої фотокамери і записувати їх у задану папку на зовнішню пам'ять мобільного пристрою.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

У разі успішного виконання лабораторної роботи, студент буде вміти створювати мобільні додатки з деяким набором функцій.

КОНТРОЛЬНІ ЗАПИТАННЯ

Створення нового проекту в Eclipse SDK

Структура файлу AndroidMonifest.xml.

Які функції застосовуються при управлінні камерою?

Функції для роботи з файлами.

МЕТОДИЧНІ МАТЕРІАЛИ ТА ВКАЗІВКИ

http://blogs.kpi.kharkov.ua/v2/gamehub/download

11.5 ТЕМА 3: РОЗРОБКА МУЛЬТИАГЕНТНОЇ СИСТЕМИ

АНОТАЦІЯ

Лекція знайомить з прикладом реалізації трирівневої системи управління БПЛА для реалізації мультиагентної взаємодії.

МЕТА ЛЕКЦІЇ

Ознайомити студентів з методами реалізації систем управління з використанням МАС. Розглянути питання: мульти-агентний підхід і трирівнева система управління БПЛА, реалізація БПЛА-агентів для автономної групи, алгоритми групової роботи мережі БПЛА, програмування мікрокомп'ютерів БПЛА.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

Сформувати у студентів знання щодо основних методів реалізації систем управління з використанням мультиагентних систем, вміння реалізації агентів для автономної групи, програмування обладнання агента та реалізація алгоритма групової роботи мережі агентів.

КОНТРОЛЬНІ ЗАПИТАННЯ

Вибір задач для одного та групи агентів.

Стратегія взаємодії агентів.

Визначення моделі мережі взаємодії агентів.

Переваги використання групи мультиагентів порівняно з одним агентом.

Трирівнева система управління агентом.

Призначення та функції верхнього рівня управління БПЛА-агента.

Призначення та функції середнього рівня управління БПЛА-агента.

Контури управління та їх функції.

Алоритм взаємодії БПЛА-агентів у групі з єдиним центром управління.

МЕТОДИЧНІ МАТЕРІАЛИ ТА ВКАЗІВКИ

http://blogs.kpi.kharkov.ua/v2/gamehub/download

11.6 ЛАБОРАТОРНА РОБОТА №3. ЗНАЙОМСТВО С JADE

АНОТАЦІЯ

Лабораторна робота орієнтована на оволодіння студентами навичок роботи з мультиагентною платформою JADE.

МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Знайомство з мультиагентною платформою JADE, встановити платформу на робочий комп'ютер, освоїти протокол передачі даних між агентами на платформі, запуск головного контейнера платформи та нового агента на прикладі агентів «PingAgent» запит-відповідь.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

У разі успішного виконання лабораторної роботи, студент буде вміти створювати агентні системи з використанням мультиагентної платформи JADE.

КОНТРОЛЬНІ ЗАПИТАННЯ

Процес установки JDK та JADE.

Призначення і властивості контейнера в MAC.

Створення нового Java проекту Eclipse IDE.

Підключення бібліотеки Jade в проект.

Конфігурація запуску агента.

Формування запиту до агента.

МЕТОДИЧНІ МАТЕРІАЛИ ТА ВКАЗІВКИ

http://blogs.kpi.kharkov.ua/v2/gamehub/download

11.7 ЛАБОРАТОРНА РОБОТА № 4. МУЛЬТИАГЕНТИЙ ДОДАТОК З JADE-АГЕНТОМ НА ПЛАТФОРМІ ANDROID

АНОТАЦІЯ

Лабораторна робота орієнтована на оволодіння студентами порядком та послідовністю роботи з мультиагентною платформою JADE при створеенні Android додатків.

МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Створити кілька робочих контейнерів у мультиагентній платформі JADE, визначити взаємозв'язок між кількома контейнерами, перенести один з контейнерів на мобільний пристрій, запустити додаток в Android emulator, запуск DummyAgent на мобільному пристрої з ОС Android.

ОЧІКУВАНІ РЕЗУЛЬТАТИ

У разі успішного виконання лабораторної роботи, студент буде вміти створювати додатки для ОС Android з використанням мультиагентної платформи JADE.

КОНТРОЛЬНІ ЗАПИТАННЯ

Налаштування JAVA_HOME.

Етапи та налаштування віртуального пристрою.

Запуск агента.

Формування запиту до агента.

Методичні матеріали та вказівки

http://blogs.kpi.kharkov.ua/v2/gamehub/download

12 СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Граничин О. Н. Информационные технологии в управлении / О. Н. Граничин, В. И. Кияев. — Москва : Изд–во «Бином», 2008. — 336 с.
2. Каляев И. А. Распределенные системы планирования действий коллективов роботов / И. А. Каляев, А. Р. Гайдук, С. Г. Капустян. — Москва : Янус–К, 2002. — 292 с.
3. Android for Programmers: An App-Driven Approach / Paul J. Deitel, Harvey M. Deitel, Abbey Deitel, Michael Morgano. — 1st ed. — Prentice Hall, 2011. — 512 p. — ISBN 978-0-13-212136-1.
4. Baxter J. W. Fly–by–Agent: Controlling a Pool of UAVs via a Multi–Agent System / Jeremy W. Baxter, Graham S. Horn, Daniel P. Leivers // The 27th SGAI International Conference on Artificial Intelligence. — 2008. — Vol. 21, no. 3. — P. 232–237.
5. Intentions in Communication / Ed. by Philip R. Cohen, Jerry Morgan, Martha E. Pollack. — MIT Press, 1990. — 508 p.
6. Pennebaker W. B. JPEG: Still Image Data Compression Standard / William B. Pennebaker, Joan L. Mitchell. — Kluwer Academic Publishers, 2004. — 638 p. — ISBN 978–0442012724.
7. Russell S. J. Artificial Intelligence: A Modern Approach / S. J. Russell, P. Norvig. — Prentice Hall, 2003. — 1152 p. — ISBN 0–13–790395–2.
8. Shannon C. E. A Mathematical Theory of Communication / C. E. Shannon // Bell System Technical Journal. — 1948. — Vol. 27. — P. 379–423.
9. Shoham Y. Multiagent Systems: Algorithmic, Game–Theoretic and Logical Foundations / Shoham Y., Leyton–Brown K. — London : Cambridge University Press, 2009. — 513 p.
10. Wiener N. Cybernetics: or Control and Communication in the Animal and the Machine / N. Wiener. — Cambridge : MIT Press, 1948.

Лекції

ЛЕКЦІЯ 1. МУЛЬТИАГЕНТНІ ТЕХНОЛОГІЇ МУЛЬТИАГЕНТНІ ТЕХНОЛОГІЇ . ПОНЯТТЯ «АГЕНТА».

1.1 ВСТУП

Під мультиагентними технологіями зараз часто розуміють як технології розробки і використання мультиагентних систем (МАС), так і мультиагентного управління (МАУ).

Завдання управління і розподіленої взаємодії в мережах динамічних систем привертають в останнє **десятиліття** *увагу* все більшого числа дослідників. Багато в чому це пояснюється широким застосуванням мультиагентних систем в різних областях, включаючи автоматичне підстроювання параметрів нейронних мереж розпізнавання, управління формаціями, роїння, розподілені сенсорні мережі, управління перевантаженням в мережах взаємодії груп БПЛА, відносне вирівнювання груп супутників, управління рухом груп мобільних роботів, синхронізації в енергосистемах і інше.

На практиці все частіше використовуються розподілені системи, що виконують певні дії паралельно, для яких актуальна задача поділу пакету завдань між декількома обчислювальними потоками (пристроями). Подібні завдання виникають не тільки в обчислювальних мережах, але також і у виробничих мережах, мережах обслуговування, транспортних, логістичних мережах та інших. Виявляється, що при природних обмеженнях на зв'язки, децентралізовані стратегії здатні ефективно вирішувати такого типу завдання.

1.2 МУЛЬТИАГЕНТНИЙ ПІДХІД

В основі мультиагентного підходу лежить поняття мобільного програмного агента, який реалізований і функціонує як самостійна спеціалізована комп'ютерна програма чи елемент штучного інтелекту.

Спочатку, до появи відповідних інформаційних технологій, «агент» був людиною, яка делегувала частину повноважень – як у виконанні конкретних функцій, так і в прийнятті рішень. У перших (не комп'ютерних) мультиагентних системах агенти представляли співробітників компаній, від імені і за дорученням яких, вони взаємодіяли між собою при виконанні певного завдання – наприклад, представники покупця і продавця в торговій мережі або в інших видах бізнесу. Такі системи успадковували багато рис «бюрократичної» організації, включаючи централізацію управління, статичну структуру і вузькоспеціалізовану агентну функціональність. Зокрема, базовий агент (резидент) отримував завдання, декомпозірував їх і розподіляв підзадачі між іншими агентами, після чого отримував результат і ухвалював рішення – при цьому, як правило, більшість агентів займалися виключно збором і постачанням інформації .

На зміну таким системам, що копіюють централізовану ієрархію, швидко прийшли розподілені системи, в яких знання і ресурси розподілялися між досить «самостійними» агентами, але зберігався загальний орган командного управління, який приймає

рішення в критичних або конфліктних ситуаціях. Подальшим кроком в цьому напрямку стала парадигма повністю децентралізованих систем, в яких управління відбувається тільки за рахунок локальних взаємодій між агентами. При цьому вузька функціональна орієнтація агента, на рішення якої однієї окремої частини «загальною» завдання, поступово стала поступатися місцем універсальної цілісності (автономності). Прикладами таких децентралізованих організацій частково можуть служити колонії комах, наприклад, бджіл або мурах.

Суть мультиагентних технологій полягає в принципово новому методі вирішення завдань. На відміну від класичного способу, коли проводиться пошук деякого чітко визначеного (детермінованого) алгоритму, що дозволяє знайти найкраще вирішення проблеми, в мультиагентних технологіях рішення виходить автоматично в результаті взаємодії безлічі самостійних цілеспрямованих програмних модулів – так званих агентів.

Найчастіше класичні методи вирішення завдань або непридатні до реального життя (не важко уявити собі, що значить спробувати вирішити задачу управління підприємством в непередбачуваній динамічній обстановці сучасного бізнесу, навіть за допомогою вищої математики), або вони вимагають величезних обсягів розрахунків (для яких не вистачить потужності всіх сучасних комп'ютерів), або вони зовсім відсутні.

Чи означає це, що ситуація, коли точний алгоритм рішення відсутній, безнадійна?

Ні, відповідають мультиагентні технології. Зрештою, людям у своєму житті постійно доводиться в умовах дефіциту часу і коштів вирішувати завдання, які не мають точного формального рішення – і вони вирішуються часто не найгіршим чином.

На рис. 1.1 показані в порівнянні дві схеми побудови програмного забезпечення: традиційна і на базі мультиагентної системи. У МАС кожної сутності ставиться у відповідність програмний агент, який представляє її інтереси.

Традиційна схема побудови програмної системи

Мультиагентна система

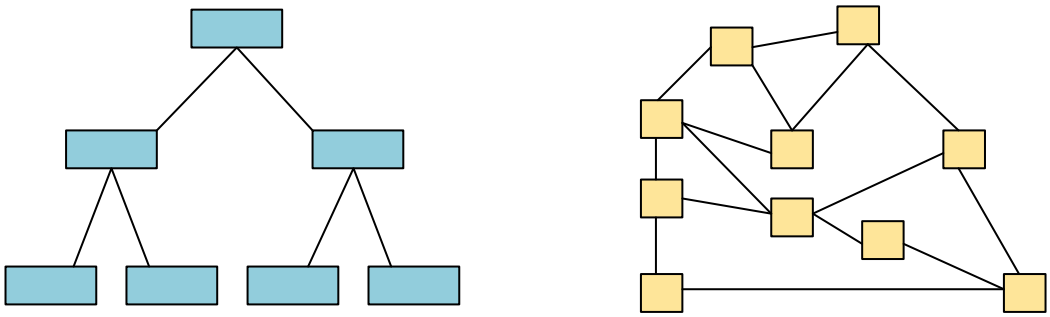

Рис. 1.1. Схеми побудови програмного забезпечення

Справа в тому, що людині властивий інтелект – це його відрізняє від комп'ютера, який діє виключно по закладеній у нього програмі. Те, що дозволяє йому орієнтуватися в складній обстановці, мати справу з нечітко поставленими завданнями, адаптуватися до мінливих умов. Невизначеність присутня, коли існує набір альтернатив,

і неможливо передбачити, який з варіантів виявиться кращим після досить тривалого часу.

При складанні розкладів руху вантажівок, це та ситуація, коли, наприклад, існує вибір між декількома вантажівками, які перевозять вантажі, декількома шляхами, які можуть бути використані для досягнення різних точок призначення, і багатьма водіями, які можуть управляти вантажівками. Кожен з ресурсів (вантажівка, дорога і водій) мають різні властивості.

Невизначеність зростає в ситуаціях, коли можливі непередбачувані події, такі як зміни в умовах поставок або попиту, аварії або збої ресурсу, затримки, скасування замовлень, тощо.

А чи є інтелект, скажімо, у колонії мурах?

З одного боку, кожен окремо взятий мураха, очевидно, ним не володіє. З іншого – колонія в цілому проявляє дивовижні зразки поведінки, яка багато в чому може вважатися інтелектуальною. Такі ситуації називаються проявом Емерджентні інтелекту, або несподіваних властивостей, якими володіє система, але не володіє жоден окремий елемент, котрий належить до неї. Виникає при цьому ефект «інтелектуального резонансу» часто так і називають «Інтелект рою». Дійсно, інтелект і фізична сила однієї бджоли не такі великі, але рій бджіл, котрий діє узгоджено, може перемогти ведмедя і навіть людину. Агенти дуже схожі на членів команди, які можуть змагатися один з одним або співпрацювати в процесі прийняття рішення. Ключова особливість Емерджентні інтелекту – динаміка і непередбачуваність процесу прийняття рішень. На практиці це означає, що рішення досягається за рахунок сотень і тисяч взаємодій, які майже неможливо відстежити. Але це і не потрібно, оскільки агентам дають цілі, які вони повинні досягати, але не визначають сценарії виконання завдань по досягненню цих цілей. Ці сценарії формуються і виконуються агентами самостійно. На кожному кроці агенти розглядають входи системи і реагують на непередбачувані події (затримки, збої, зміни). Реакція може бути самостійною, або здійснюватися у взаємодії з оператором. Таким чином, емерджентний інтелект – це не є якийсь новий або спеціально сконструйований унікальний «блок», доданий до системи. Навпаки, це щось (результат самоорганізації), що виникає як би «з повітря» (за рахунок безлічі прихованих або явних умов, що склалися в ситуації), спонтанно і в заздалегідь непередбачений момент часу, і так само несподівано зникає, але в процесі свого існування визначальним чином керує роботою всієї системи. Тут ми маємо справу з виникненням порядку з хаосу, з одним з тих явищ, які вивчали і описували такі видатні вчені, як Олександр Богданов (теорія організації), Ілля Пригожий (самоорганізація в фізичних системах), Марвін Мінський (психологія і теорія мислення), Артур Кестлер (біологія).

1.3 МУЛЬТИАГЕНТНІ СИСТЕМИ

На початку XXI ст. група провідних світових вчених, пропрацювавши кілька років, склала список пріоритетних завдань кібернетики на найближчі 50 років. Серед них:

- динамічно реконфігуроване інтелектуальне управління,
- асинхронна теорія управління,
- управління через Інтернет,
- перепрограмування системи управління бактеріями,
- створення футбольної команди роботів, яка виграє у переможця кубка світу серед людей.

МАС кардинально відрізняються від традиційних «жорстко» організованих систем, і, в перспективі, здатні допомогти у вирішенні цих завдань.

Початок побудови моделей і застосування штучних мультиагентних систем (МАС) на практиці було покладено в 1960 – х роках. В якості основи були взяті досягнення таких областей діяльності людини, як системи штучного інтелекту (Artificial Intelligence), паралельні обчислення (Parallel Computing), розподілене рішення задач (Distributed Problem Solving). Багатоагентні системи мають реальну можливість інтегрувати в собі самі передові досягнення перерахованих областей, демонструючи принципово нові якості. Зараз МАС – один із напрямків, котрий розвивається найбільш динамічно і є одним із перспективних напрямків в області штучного інтелекту.

Відкритий характер сучасного інформаційного суспільства та глобальної ринкової економіки призводить до прискорення науково-технічного прогресу і загострення конкуренції на ринках. Це змушує підприємства шукати нові методи і засоби організації і управління, спрямовані на більш якісне і ефективне задоволення індивідуальних запитів споживачів. Більшість сучасних систем характеризуються відсутністю коштів своєчасної ідентифікації нових потреб і можливостей в середовищі, що дозволяє підприємству оперативно приймати ефективні рішення по реконфігурації виробничих, кадрових, фінансових та інших ресурсів. Типовими прикладами подій, що викликають необхідність заново ідентифікувати потреби та можливості, є: поява нового вигідного замовлення, для виконання якого недостатньо власних ресурсів підприємства, вихід з ладу частини наявних ресурсів, а також зміна критеріїв прийняття рішень. Чим вище невизначеність, тим більш розподілений характер мають процеси прийняття рішення і чим частіше трапляються незаплановані події, тим нижче ефективність існуючих систем, нездатних самостійно приймати рішення і автоматично перебудовуватися під зміни в середовищі. Крім того, необхідність модифікації схеми прийняття рішень в традиційних системах виявляється складним і трудомістким завданням, котре вимагає високої кваліфікації виконавців. Це робить розробку і експлуатацію таких систем вкрай дорогими. Відповідно, ще однією актуальною проблемою сьогодення стає зростання обсягів інформації і ступеня складності опису систем.

Для вирішення подібних проблем застосовуються мультиагентні технології, в основі яких лежить поняття «агента», яке останнім часом було адаптовано до багатьох областей, як прикладного і системного програмування, так і до досліджень в областях штучного інтелекту і розподілених інтелектуальних систем. Причому в кожному конкретному випадку поняттю надається дещо різне значення.

1.4 АГЕНТИ

Спочатку ідея створення інтелектуального посередника «агента», виникла у зв'язку із бажанням спростити стиль спілкування кінцевого користувача з комп'ютерними програмами, оскільки домінуючий, в основному, і нині стиль взаємодії користувача з комп'ютером, коли користувач запускає завдання явно і управляє його рішенням. Але це абсолютно не підходить для недосвідченого користувача. Інакше кажучи, спочатку ідея інтелектуального посередника виникла як спроба інтелектуалізації призначеного для користувача інтерфейсу.

Розвиток методів штучного інтелекту дозволив зробити новий крок до зміни стилю взаємодії користувача з комп'ютером. Виникла ідея створення так званих «автономних агентів», які породили вже новий стиль взаємодії користувача з програмою. Замість взаємодії, ініційованої користувачем шляхом команд і прямих маніпуляцій, користувач втягується в спільний процес вирішення. При цьому, як користувач, так і комп'ютер, обидва беруть участь в запуску завдання, управлінні подіями і вирішенні задачі. Для такого стилю використовується метафора персональний асистент, який співпрацює з користувачем в цьому ж робочому середовищі.

Словники дають таке тлумачення слова агент: «хтось або щось, що прикладає зусилля для досягнення ефекту». Таке саме загальне визначення вказує на першу ознаку агента – агенти здійснюють дії. Існує твердження, що агенти не просто здійснюють дії, але вони діють автономно і раціонально. Під автономністю зазвичай розуміють, що агент діє без прямого втручання людини або іншої керуючої суті. Під раціональністю розуміють прагнення агента оптимізувати значення деякої оціночної функції. Міра раціональності неявно вказує на те, що агент має мету (бажання англ. desires), яку агент «хоче» досягти, і уявлення про навколишній світ (переконання, англ. beliefs), на які агент спирається при виборі дії (реалізації намірів, англ. intentions – безліч обраних, сумісних і досяжних бажань).

Ще однією важливою властивістю агента є те, що він поміщений під зовнішнє середовище, з якого він здатний взаємодіяти. Зазвичай, середовище не контролюється агентом, він лише здатний впливати на нього. Поділ намірів і бажань необхідне, так як агент може мати несумісні бажання або бажання можуть бути недосяжні. Оскільки агент обмежений в ресурсах і не може досягти всіх бажань одночасно, природно вибирати найбільш значущі цілі – наміри. Отже, агент – розумна сутність, поміщена в зовнішнє середовище, здатна взаємодіяти з нею, роблячи автономні раціональні дії для досягнення цілей, тобто інтелектуальний агент – це агент, що володіє наступними властивостями:

- реактивність – агент відчуває зовнішнє середовище і реагує на зміни в ньому, здійснюючи дії, спрямовані на досягнення цілей;
- проактивність – агент показує керовану цілями поведінку, проявляючи ініціативу, здійснюючи дії спрямовані на досягнення цілей;
- соціальність – агент взаємодіє з іншими сутностями зовнішнього середовища (іншими агентами, людьми і т. п .) для досягнення цілей.

При розробці системи кожна з перших двох властивостей досягається досить легко. Найбільшу складність представляє поєднання в системі обох властивостей в потрібних пропорціях. Буде не дуже ефективно, якщо агент «жорстко» слідуватиме за сценарієм досягнення мети, не реагуючи на зміни у зовнішньому середовищі і не володіючи здатністю помітити необхідність коригування плану. Але також неефективним буде і поведінка, обмежена лише реакцією стимули, котрі поступають із-зовні, без будь-якого планування цілеспрямованих дій. Насправді, описана проблема настільки складна, що навіть далеко не всі люди здатні ефективно її вирішувати. Дуже часто можна побачити людину, котра кидається на кожну можливість, що підвернулася, але ніколи не доводить нічого до кінця, тобто не концентрується на цій можливості достатній час, щоб повноцінно її реалізувати. Але також часто зустрічаються люди, які, одного разу поставивши собі за мету і сформувавши план, будуть намагатися принципово йому слідувати, не помічаючи змін в ситуації, що вимагають перегляду цілей та планів.

Досягти властивості соціальності теж нелегко. Соціальність – це не просто обмін даними. Крім комунікації, соціальна поведінка має включати кооперацію з іншими сутностями, яка полягає в поділі цілей між окремими сутностями, спільному плануванні та координації дій, спрямованих на досягнення загальних цілей. Соціальна поведінка, як мінімум, передбачає наявність у агента уявлень про цілі інших сутностей і те, як вони планують цих цілей досягти.

Складність формулювання змістовних практично значущих завдань і неможливість точного завдання всіх умов функціонування висувають адаптивні постановки проблем, окремо виділяючи таку особливість агентів, як адаптивність – здатність автоматично пристосовуватися до невизначених і мінливих умов в динамічному середовищі.

Таким чином, попередниками програмних агентів можна вважати складні адаптивні системи, які вміють підлаштовуватися під ситуацію або обставини і принциповим чином змінювати свою поведінку або характеристики, щоб забезпечити вирішення поставлених перед ними завдань. Однак у випадках, коли агент функціонує в складному, постійно мінливому середовищі, взаємодіючи при цьому з іншими агентами, така мультиагентна система значно складніша просто адаптивної системи, так як вона швидше навчається і може діяти ефективніше за рахунок перерозподілу функцій або завдань між агентами.

Складні системи часто розглядають як середовище дії агентів. З поняттям складних систем пов'язані наступні фундаментальні ідеї, які безпосередньо впливають на функціонування MAS:

- в складних системах існують автономні об'єкти, які взаємодіють один з одним при виконанні своїх визначених завдань;
- агенти повинні мати можливість реагувати на мінливі умови середовища, в якій вони функціонують і, можливо, змінювати свою поведінку на основі отриманої інформації;
- складні системи характеризуються виникненням логічно пов'язаних структур, які формуються в результаті взаємодії між агентами;
- складні системи з виникаючими структурами часто існують на межі порядку і хаосу;

- при створенні складних систем на базі агентів має сенс розглядати біологічні аналогії, такі як: паразитизм, симбіоз, репродукцію, генетику, мітоз і природний відбір (наприклад, компанія British Telecom при формуванні мережі напрямків дзвінків використовує модель діяльності колонії мурах).

Концепція агентів, розроблена в рамках мультиагентних технологій і мультиагентних систем (MAS), передбачає наявність активної поведінки агентів, тобто здатності комп'ютерної програми самостійно реагувати на зовнішні події і вибирати відповідну реакцію. Сьогодні агентні технології пропонують різні типи агентів, моделі їх поведінки і властивості, сімейство архітектур і бібліотеки компонентів, орієнтовані на сучасні вимоги.

В даний час не існує усталеного визначення агента. Нижче перераховані деякі з них:

«Агент – це апаратна або програмна сутність, здатна діяти в інтересах досягнення цілей, поставлених користувачем».

«Під агентом можна розуміти самостійну програмну систему, що складається з програм–об'єктів, що має можливість приймати вплив з зовнішнього світу, визначати свою реакцію на цей вплив і відповідно до цього формувати відповідну дію. Такі агенти здатні діяти, «міркувати» і обмінюватися даними один з одним в мережі для формування індивідуальних чи колективних рішень».

За визначенням Крістіана Доннегара (директора по технології компанії Living Systems, що займається створенням систем спільної комерції на основі технології агентів): «агенти – програмні об'єкти, які виконують певні попереджувальні та коригувальні дії відповідно до завдань, що делеговані їм людиною».

Алан Кей, який почав першим розвивати теорію агентів, визначив агента як «програму, яка після отримання завдання здатна поставити себе на місце користувача і діяти за адаптивним сценарієм. Якщо ж агент потрапляє в глухий кут, він може задати користувачеві питання, щоб визначити, яким чином йому необхідно діяти далі».

Проста комп'ютерна програма відрізняється від агента тим, що «не обтяжує» себе цільовою поведінкою і аналізом досягнутих результатів. Навпаки, агент, що представляє інтереси користувача, «зацікавлений» у тому, щоб завдання було виконано. У разі невдачі або якогось збою, він повинен повторити спробу пізніше або мати про запас альтернативний варіант вирішення проблеми. Агенти в процесі відпрацювання завдань завжди формують список виконаних дій, результати тестування і верифікації і відсилають його в керуючу систему .

Відзначимо, однак, що питання щодо визначення того, що таке агент не закрите досі, і обговорення цього питання періодично виноситься на конференції найвищого рівня. На рис. 1.2 показані області знання і технології, за допомогою яких формуються механізми штучного інтелекту і застосування мультиагентних систем.

На підставі викладеного вище, можна скомпілювати наступне визначення: агент – це самостійна програмна система, котра:

- має можливість приймати вплив із зовнішнього світу;
- визначає свою реакцію на цей вплив і формує відповідну дію;

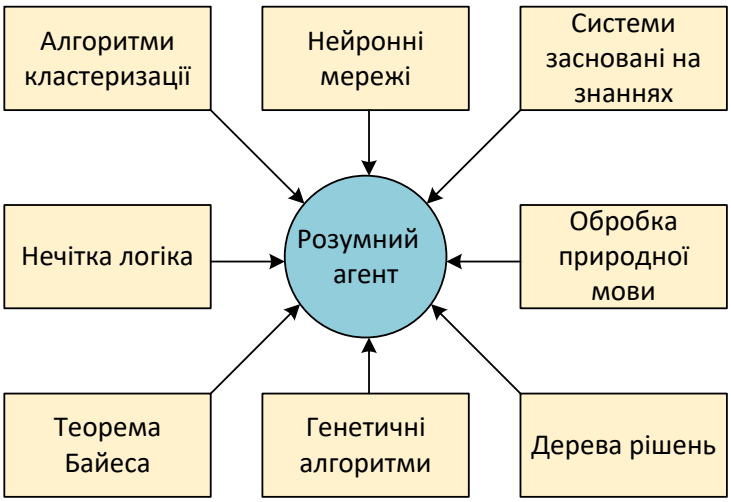

Рис. 1.2. Області знання і технології, які використовуються інтелектуальними агентами.

- змінює свою поведінку з плином часу в залежності від накопиченої інформації і отриманих з неї знань;
- володіє мотивацією і здатна після делегування повноважень користувачем поставити себе на його місце і прийняти рішення, відповідне ситуації.

Інтелектуальний агент повинен мати наступні властивості:

- автономність – здатність функціонувати без втручання з боку свого власника і здійснювати контроль внутрішнього стану і своїх дій;
- адаптивність – агент має здатність навчатися;
- коллаборативного – агент може взаємодіяти з іншими агентами декількома способами, граючи різні ролі;
- здатність до міркувань – агенти можуть володіти частковими знаннями або механізмами висновків, а також спеціалізуватися на конкретну предметну область;
- комунікативність – агенти можуть спілкуватися з іншими агентами;
- мобільність – здатність передачі коду агента з одного сервера на інший;
- соціальну поведінку – можливість взаємодії і комунікації з іншими агентами;
- реактивність – адекватне сприйняття середовища і відповідні реакції на її зміни;
- активність – здатність генерувати цілі і діяти раціонально для їх досягнення;
- наявність базових знань – знання агента про себе, навколишнє середовище, включаючи інших агентів, які не змінюються в рамках життєвого циклу агента;
- наявність переконань – змінна частина базових знань, які можуть змінюватися в часі;
- наявність мети – сукупність станів, на досягнення яких спрямована поточна поведінка агента;
- наявність бажань – стану і/або ситуації, досягнення яких для агента важливе;
- наявність зобов'язань – завдання, які бере на себе агент на прохання і/або за дорученням інших агентів;
- наявність намірів – те, що агент повинен робити в силу своїх зобов'язань/або бажань.
- Іноді в цей же перелік додаються і такі людські якості, як раціональність, правдивість, доброзичливість.

ЛЕКЦІЯ 2. РОЗРОБКА МУЛЬТИАГЕНТНИХ СИСТЕМ ВЗАЄМОДІЯ АГЕНТІВ. ПЛАТФОРМИ ДЛЯ РОЗРОБКИ МУЛЬТИАГЕНТНИХ СИСТЕМ. ПРИКЛАДИ РЕАЛІЗАЦІЙ

2.1 МУЛЬТИАГЕНТНІ СИСТЕМИ

Система, у якій декілька агентів можуть спілкуватися, передавати одне одному деяку інформацію, взаємодіяти між собою і розв'язувати поставлене завдання, називається мультиагентною системою (МАС). У МАС завдання (або підзавдання) розподілені між агентами, кожен з яких розглядається як член групи або організації. Розподіл завдань передбачає призначення ролей кожному з членів групи, визначення міри його «відповідальності» і вимог до «досвіду».

Системи з багатьма агентами з'явилися на перетині теорії систем і розподіленого штучного інтелекту. З одного боку, йдеться про відкриті, активні системи, такі системи, що розвиваються, та в яких головна увага приділяється процесам взаємодії агентів як причин виникнення системи з новими якостями. З іншого боку, досить часто МАС будуються як об'єднання окремих інтелектуальних систем, що засновані на знаннях. МАС зазвичай складається з таких основних компонентів:

- множина організаційних одиниць, у якій виділяються підмножини агентів, що маніпулюють підмножинами об'єктів;
- множина завдань;
- середовище – тобто деякий простір, в якому існують агенти і об'єкти;
- множина відносин між агентами;
- множина дій агентів (наприклад, множина операцій над об'єктами).

Основною формою організації взаємодії між агентами, що характеризується об'єднанням їхніх зусиль для досягнення спільної мети при одночасному розподілі між ними функцій, ролей і обов'язків, є кооперація. У загальному випадку це поняття можна визначити формулою:

кооперація = співпраця + координація дій + вирішення конфліктів.

Під координацією зазвичай розуміють управління залежностями між діями. Комунікація між штучними агентами залежить від обраного протоколу, що представляє собою множину правил, що визначають, як синтезувати значущі і правильні повідомлення. Фундаментальними особливостями групи, складеної з агентів, що співпрацюють для досягнення спільної мети, є соціальна структура і розподіл ролей між агентами.

Основою архітектури агента є контекст, або серверне середовище, в якому він виконується. Кожен агент має постійний ідентифікатор – ім'я. У серверному середовищі може виконуватися не тільки вихідний агент, а й його копія. Агенти здатні самостійно створювати свої копії, розсилаючи їх по різних серверах для виконання роботи. Після прибуття агента на наступний сервер, його код і дані переносяться у новий контекст і знищуються на попередньому місцезнаходженні. У новому контексті агент може робити все, що там не заборонено. Після закінчення роботи в контексті агент може пе-

реслати себе в інший контекст або на вихідну адресу відправника. Агенти здатні також вимикатися («вмирати») самі або за командою сервера, який переносить їх після цього з контексту в місце, призначене для зберігання.

На рис. 2.1 показано укрупнену структуру типового агента. Входами є внутрішні параметри агента і дані про стан середовища. Виходи – параметри, що впливають на середовище і інформують користувача (або програму, що виконує роль менеджера в системі) про стан середовища та прийняті рішення. Вирішувач – процедура прийняття рішень. Вирішувач може бути як досить простим алгоритмом, так і елементом системи штучного інтелекту.

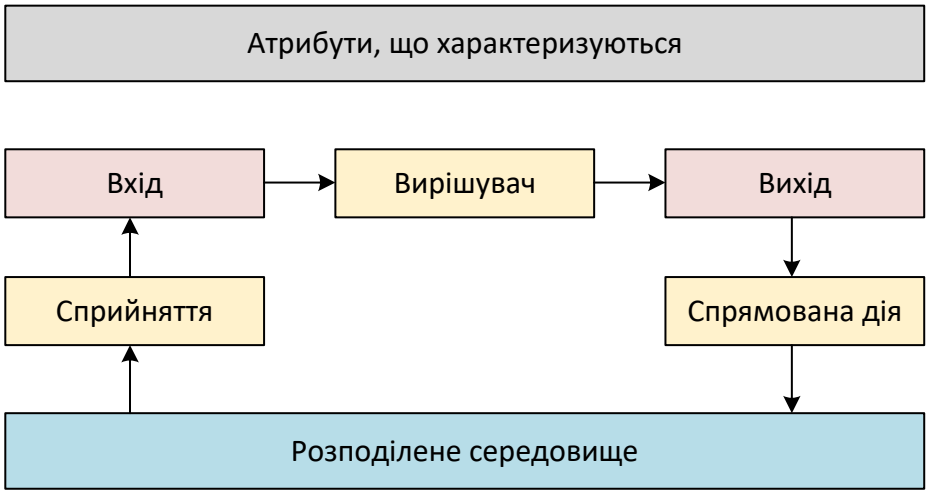

Рис. 2.1. Укрупнена структура агента

В архітектурі МАС основну частину складає предметно-незалежне ядро, в складі якого виділяються наступні базові компоненти (рис. 2.2):

- служба прямого доступу – забезпечує безпосередній доступ до атрибутів агентів;
- служба повідомлень – відповідає за передачу повідомлень між самими агентами, а також між агентами і додатковими системами ядра;
- бібліотека класів агентів (частина бази знань) – містить інформацію про класифікацію агентів в даній МАС;
- спільнота агентів – серверне «місце», де розміщуються агенти; цей блок, окрім життєдіяльності агентів, забезпечує ще функції по завантаженню/запису агентів і їх властивостей і оптимізацію роботи агентів з ресурсами;
- онтологія – предметна база знань, яка містить конкретні знання про об'єкти, що подаються у вигляді відповідної семантичної мережі.

Загальну методологію еволюційного проектування МАС, що здіймається, може бути представлено ланцюжком: <середовище – функції МАС – ролі агентів – відносини між агентами – базові структури МАС – модифікації>, і включає наступні етапи:

- формулювання призначення (мета, цілі розробки) МАС;
- визначення основних і допоміжних функцій агентів в МАС;
- уточнення складу агентів і розподіл функцій між агентами, вибір архітектури агентів;
- виділення базових взаємозв'язків (взаємовідносин) між агентами в МАС;
- визначення можливих дій (операцій) агентів;
- аналіз реальних поточних або передбачуваних змін зовнішнього середовища.

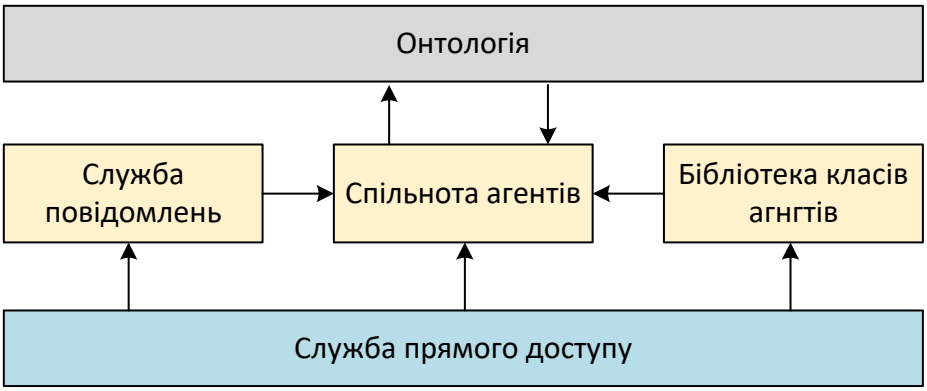

Рис. 2.2. Архітектура ядра мультиагентної системи

При проектуванні організацію агентів можна розглядати як набір ролей, які перебувають між собою у певних відносинах і взаємодіють одне з одним. Таким чином, методологія проектування МАС, що здіймається, вимагає попереднього завдання вихідних функцій (ролей агентів), визначення кола їх зобов'язань відносно інших, формування вихідних структур і структур, що розвиваються на основі виділених функцій і дослідження відповідності цих структур характеру вирішуваних завдань у виділених проблемних областях.

Головна ідея спадного проектування полягає у визначенні загальних соціальних характеристик МАС за деяким набором критеріїв, побудові базових типів їх організацій з подальшим визначенням вимог до архітектури агентів. Коли мова йде про «вирощування» штучних соціальних систем і спільнот, на перший план виходить спадний підхід до організаційного проектування.

У розглянутих нижче прикладах, безумовно, найбільш підходящим є проектування, що здіймається.

2.1 СУЧАСНІ МІЖНАРОДНІ СТАНДАРТИ СТВОРЕННЯ АГЕНТІВ І ПЛАТФОРМИ МАС

Існує декілька міжнародних підходів до створення мультиагентних систем, найбільш відомі з яких – це OMG MASIF, створений Object Management Group, в основі якого лежить поняття «мобільний агент»; специфікації FIPA (Foundations for Intelligent Physical Agents), засновані на припущенні про інтелектуальність агента, а також стандарти, розроблені дослідницьким підрозділом Пентагону – Агентством передових оборонних наукових досліджень (Defense Advanced Research Projects Agency – DARPA), зокрема Control of Agent Based Systems.

Щодо мобільності і інтелектуальності агентів, більшість фахівців сходяться на тому, що мобільність – центральна характеристика агента, інтелектуальність – бажана, але не завжди суворо необхідна.

Діяльність FIPA полягає в спільному дослідженні та розробці членами організації міжнародних узгоджених специфікацій, які дозволять максимізувати взаємодію між агентними додатками, послугами та обладнанням. Членами FIPA є такі високотехнологічні компанії як Alcatel, Boeing, British Telecom, Deutsche Telekom, France Telecom, Fujitsu, Hitatchi, HP, IBM, Fujitsu, Hewlett Packard, IBM, Intel, Lucent, NEC, NHK, NTT,

Nortel, Siemens, SUN, Telia, Toshiba, різні університети, державні організації.

Специфікації FIPA орієнтуються на забезпечення можливості взаємодії інтелектуальних агентів через стандартизовану комунікацію агентів та через мови контенту. Поряд із загальними основами комунікації FIPA спеціалізується також на протоколах онтології і переговорів для підтримки взаємодії в конкретних прикладних сферах (транспортна підтримка, виробництво, мультимедіа, підтримка мережевої взаємодії).

Стандарт OMG MASIF націлений на створення умов для міграції мобільних агентів між мультиагентними системами за допомогою стандартизованих інтерфейсів CORBA IDL.

Організація DARPA ініціювала роботу з розподілу знань (Knowledge Sharing Effort), в результаті якої мови програмування агентів були розділені на синтаксис (syntax), семантику (semantics) і прагматику (pragmatics):

- KIF – Knowledge Interchange Format (syntax);
- Ontolingua – a language for defining sharable ontologies (semantics);
- KQML (Knowledge Query and Manipulation Language) – a high-level interaction language (pragmatics).

Важливим елементом при створенні мультиагентних систем є мова комунікації агентів – Agent Communication Language, що визначає типи повідомлень, якими можуть обмінюватися агенти. В рамках парадигми комунікації між агентами, кооперація між ними досягається за рахунок ACL, мови контенту і онтології, які визначають набір базових концепцій, використовуваних в повідомленнях кооперації. Онтологія тут виступає синонімом поняття API (Application Programming Interface), тобто вона визначає конкретний інтерфейс інтелектуальних агентів.

На технічному рівні комунікація між агентами відбувається за рахунок передачі повідомлень, використовуючи якийсь транспортний протокол нижнього рівня (SMTP, TCP / IP, HTTP, IIOP). Альтернативами використання ACL є ряд інших мов – мови БД (SQL), Distributed object systems (CORBA і ін.), Service languages (e – speak від Hewlett Packard, BizTalk від Microsoft і ін.) I Web languages (XML, RDF, DAML).

Ще однією альтернативою ACL є CORBA ORB, розроблений вже згадуваною Object Management Group. Вся функціональність, що надається CORBA, доступна і на мові JAVA, шляхом комбінації Java RMI, Java RMI servers, Jini, Java event servers та інших.

Зараз мови комунікації агентів продовжують еволюціонувати. Оскільки сумісність – визначальна характеристика агентів, при розробці MAS дуже важливою є саме стандартизована комунікативність. Основними об'єктами для стандартизації є: архітектура агента, мови взаємодії агентів, протоколи взаємодії агентів, знання агентів, мови програмування агентів.

Як відзначають експерти в області розробки агентів, для подальшої еволюції технологій створення агентів необхідні наступні дії:

- розвиток семантики мов комунікації агентів (ACL) (загальних мов контенту і онтології; мов для опису дій агентів, намірів і прагнень);
- розвиток онтології агентів (відокремлювані онтології для властивостей агентів і їхньої поведінки);

- поліпшення використання метаданих (абстрактне і суміщуване з багатьма мовами контенту);
- декларативні і ясні протоколи (мови для визначення протоколів високого рівня, що базуються на більш примітивних);
- практичний обмін знаннями між агентами (соціальні механізми для обміну інформацією і знаннями, розгляд обміну знаннями як мобільний код);
- розвиток схем і методів для контролю за системами агентів (штучні ринки, природний відбір і т. ін.).

Агентні платформи є одним із способів побудови розподілених систем і дають можливість окреслити і надати доступ для всіх додатків, що працюють на агентній платформі, до необхідних їм сервісів. Крім того, у функції агентної платформи входить розподіл агентів, аудит їхнього функціонування і управління.

На даний момент відомо кілька агентних платформ, орієнтованих на використання специфікації FIPA – 2000 (табл. 2.1) (Bellifemine F., 1999; Willmott SN, 2000; Burg B., 2001).

Таблиця 2.1

Компанія	Агентна платформа	Адреса в інтернеті
BTexact Technologies (Великобританія)	ZEUS	http://www.labs.bt.com/projects/agents/zeus/
Comtec (Японія)	Comtec Agent Platform	http://fipa.comtec.co.jp/glointe.htm
CSELT (Італія)	JADE	http://jade.tlab.com/
Fujitsu Labs (США)	AAP	http://www.sourceforge.net/
Nortel Networks	FIPA-OS	http://www.nortelnetworks.com/fipa-os

Агентна платформа в стандартах FIPA є такої конструкції (рис. 2.3).

Система управління агентами (СУА) являє собою також агента, який здійснює контроль доступу і використання агентної платформи. У кожній агентній платформі присутня одна СУА, що надає сервіс життєвого циклу програмних агентів та їх реєстр з ідентифікаторами, а також містить інформацію про стан кожного програмного агента. Маршрутизатором каталогу є програмний агент, який забезпечує направлення запитів в інші агентні платформи. Система транспортування повідомлень, або канал комунікації агентів, є програмним компонентом для управління потоками повідомлень, що приходять на агентну платформу.

2.2 ПЛАТФОРМИ ДЛЯ РОЗРОБКИ МАС

Перед тим, як приступати до створення додатків на Android, необхідно вибрати відповідний інструментарій розробки і встановити відповідний Android SDK.

Найбільш популярні засоби розробки МАС такі:

- JADE (Java Agent Development Framework) – широко використовуване програмне середовище для створення мультиагентних систем і додатків, що підтримує FIPA – стандарти для інтелектуальних агентів. Включає в себе середовище виконання агентів (агенти реєструються і працюють під управлінням середовища), бібліотеку класів, які використовуються для розробки агентних систем, набір графічних утиліт для адміністрування і спостереження за життєдіяльністю активних агентів. Про-

грамне середовище JADE підключається до будь-якого проекту на мові Java. Агенти JADE можуть бути абсолютно різними – від простих, що тільки реагують, до складних – ментальних.

- JACK Intelligent Agents – Java платформа для створення мультиагентних систем. Так само як і JADE, вона розширює Java своїми класами. JACK одна з небагатьох платформ, де використовується модель логіки агентів, заснована на переконаннях – бажаннях – намірах (Belief – desire – intention software model – BDI), і має вбудовані формально – логічні засоби планування роботи агентів.
- MadKIT – модульна і масштабована мультиагентна платформа, написана на Java. Підтримує агентів на різних мовах: Java, Python, Jess, Scheme, BeanSchell. Красиво візуалізує і дозволяє управляти цими агентами.
- AgentBuilder – великий комерційний продукт, що випускається також і в Academic Edition. Агенти достатньо інтелектуальні, спілкуються на мові KQML (Knowledge Query and Manipulation Language) і мають ментальну модель. Платформа є Java-орієнтованою.
- Cougaar (Cognitive Agent Architecture) – також Java-орієнтована платформа для побудови розподілених мультиагентних систем. Включає не тільки виконуючу систему (run-time engine), але і деякі засоби для візуалізації, управління даними та ін.
- NetLogo – кроссплатформенне програмоване середовище для програмування мультиагентних систем.
- VisualBots – безкоштовний мультагентний симулятор в Microsoft Excel з Visual Basic синтаксисом.
- MASON – Java бібліотека для моделювання мультиагентних систем.
- REPAST – набір інструментів для створення систем, заснованих на агентах.
- CogniTAO – C++ платформа розробки автономних мультиагентних систем, орієнтована на реальних роботів і віртуальних істот (CGF).

2.3 ОБЛАСТІ ЗАСТОСУВАННЯ

На сьогоднішній день мультиагентні системи використовуються для розробки широкого спектра інформаційних і промислових систем. У промисловості МАС найбільш поширені для вирішення завдань автоматизації управління складними системами, для збору і обробки інформації, в іграх. Мультиагентні технології застосовують в управлінні мобільними ресурсами, а також в таких сферах, як проектування об'єктів, промислове виробництво, фінансове планування та аналіз ризиків, розпізнавання образів, вилучення знань з даних, розуміння тексту і для вирішення інших складних проблем.

Так, наприклад, IBM використовує агентів для виробництва напівпровідникових мікросхем датська суднобудівна компанія – для заварювання отворів у кораблях, а в Японії система на базі агентів виконує функції інтерфейсу оператора надшвидкісного потягу.

МАС можуть застосовуватися як для конструювання і моделювання гнучких виробничих систем, так і для управління реальними системами виробництва (логістика), продажу продукції різного призначення (е – комерції), інтеграції і управління знаннями та

наукової роботи. Велике значення у мультиагентному підході має соціальний аспект вирішення сучасних завдань як його концептуальна основа. Такі системи повинні постійно «жити» на сервері підприємства і безперервно брати участь у вирішенні завдань, а не запускатися час від часу, а для цього – забезпечувати користувачу можливість введення нових даних і компонентів. Нарешті, такі системи повинні накопичувати інформацію, отримувати від неї нові знання і в залежності від цього змінювати свою поведінку з часом.

Зараз інтелектуальні агенти застосовуються в наступних областях бізнесу:

- управління розподіленими або мережевими підприємствами;
- складна і багатофункціональна логістика;
- віртуальні організації та Інтернет – портали з продажу продуктів і послуг;
- управління навчальним процесом в системах дистанційного навчання;
- компанії з розвиненими дистриб'юторськими і транспортними мережами (наприклад, в Procter & Gamble);
- управління каналами розподілу;
- моделювання побажань користувачів.

Для великих компаній переваги мультиагентного підходу очевидні. Серед них можна відзначити: скорочення термінів вирішення проблем, зменшення обсягу переданих даних за рахунок передачі іншим агентам високорівневих часткових рішень; скорочення термінів узгодження умов і формування замовлень.

Для розподілених компаній переваги в першу чергу полягають в можливості оптимального забезпечення продукцією, полегшенні контролю віддалених підрозділів і структур і взаємодії з ними.

Для компаній з широким і швидко змінюваним асортиментом – можливість гнучко реагувати на зміни в побажаннях клієнтів і прораховувати періоди змін. Для компаній надають послуги – накопичення досвіду взаємодії і вирішення проблем не тільки «у головах» співробітників, але і в МАС.

Серед прикладів комп'ютерних програм-агентів, що існують в даний час і широко використовуються в Інтернеті можна виділити такі:

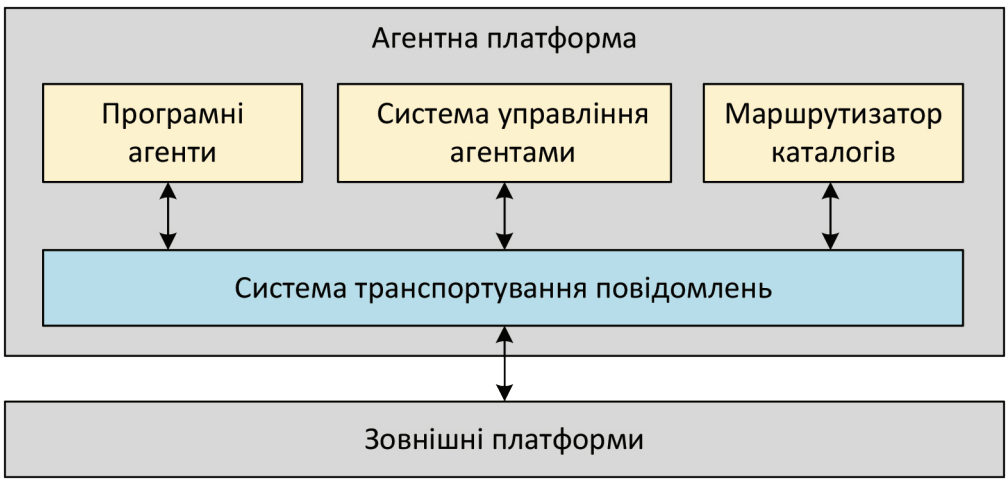

Рис. 2.3. FIPA – модель агентної платформи

Copernic Agent (http://www.copernic.com/) – одночасно відправляє запити кільком популярним пошуковим системам, вибирає найбільш рейтингові посилання, зрівнює їх між собою, видаляє дублі і, сортуючи відібране за рейтингом відповідно до свого алгоритму ранжування, виводить їх користувачеві.

MySimon (http://www.mysimon.com/) – здійснює інтелектуальний пошук, порівнюючи ціни мільйонів товарів в більш ніж двох тисячах онлайнових магазинів.

MP3 – Wolf (http://www.trellian.com/) – сканує Інтернет в пошуках потрібних користувачеві музичних файлів. У процесі роботи він використовує різні пошукові системи, а також сайти, знайдені ним раніше і ті, що містяться в його базі.

WebSite – Watcher (http://www.aignes.com/) – призначена для стеження за змінами на сайтах. Підтримує роботу RSS – стрічки. Має гнучкі настройки щодо запобігання помилкових спрацьовувань, коли окремі зміни на сторінках носять випадковий або технічний характер, наприклад зміна числа переглядів.

Крім цього, агенти можуть бути уповноваженими представниками користувача при спілкуванні з іншими користувачами або їх агентами, при вирішенні доручених їм завдань.

Для вирішення завдань автоматизації управління ресурсами підприємств в реальному часі, останнім часом розробляється велика кількість інтелектуальних програмних систем нового покоління, побудованих на основі мультиагентних технологій, які дозволяють автоматизувати повний цикл управління мобільними ресурсами в реальному часі, включаючи:

- оперативну реакцію на важливі події;
- динамічне планування і адаптивне перепланування замовлень / ресурсів;
- взаємодію з клієнтами, менеджерами і виконавцями для узгодження прийнятих рішень через Інтернет або стільниковий телефон;
- моніторинг виконання побудованих планів і бізнес-процесів замовника;
- перепланування розкладів у разі неузгодженості між планом і фактом.

У таких системах агенти здатні взаємодіяти один з одним шляхом переговорів і демонструвати колективний інтелект, що виникає в системі у формі спонтанних ланцюжків узгоджених змін планів агентів.

Таким чином, в розробці програмних систем використовуються фундаментальні принципи самоорганізації і еволюції, властиві живим системам, наприклад, колонії мурах або рою бджіл, що відрізняються здатністю вирішувати складні завдання в реальному часі, відкритістю до змін, високою ефективністю, надійністю і живучістю.

Зазвичай такі системи легко інтегруються з існуючими комунікаторами, обліково-контрольними системами підприємства, електронними картами, засобами GPS-навігації, RFID – чіпами і т. п.

Застосування мультиагентних інструментальних засобів дозволяє вирішити ряд складних завдань виробничої та транспортної логістики. В промислові реалізації декількох інтелектуальних систем, заснованих на мультиагентних технологіях: для ком-

паній (організовано узгоджене планування парку вантажівок для менеджерів центру і філій), для компанії корпоративного таксі (планування і розподіл поточних замовлень в режимі реального часу), для аеропорту (система управління наземними сервісами аеропорту на основі РФІД технологій). Для вирішення цих завдань використовувалася інструментальна платформа, яка складається з основних модулів, представлених на рис. 2.4.

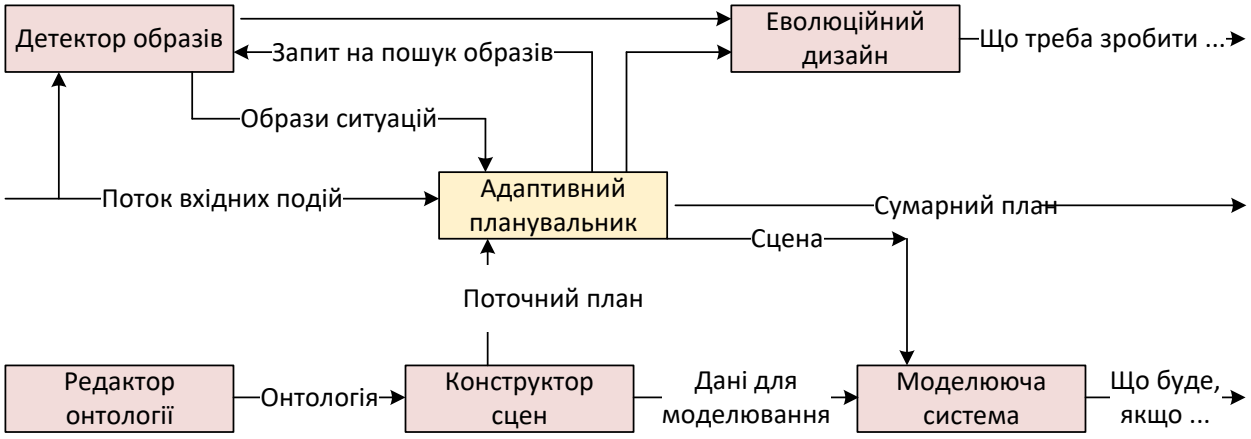

Рис. 2.4. Схема взаємодії модулів інструментальної системи

Детектор образів розпізнає типові ситуації, що виникають в ході надходження заявок і виробляє прогноз заявок і рекомендації з планування з урахуванням передісторії. Адаптивний планувальник обробляє потік вхідних подій (надходжень заявок, введення нових ресурсів, виходу з ладу ресурсів і т. п.).

Конструктор сцени дозволяє редагувати початкову конфігурацію мережі і визначити всі параметри ресурсів компанії. Конструктор сцени ґрунтується на загальній базі знань (онтології), що описує діяльність компанії, в якій присутні базові поняття і відносини між ними, і яка при розвитку бізнесу може розширюватися з використанням редактора онтології.

Редактор онтології дозволяє ввести і змінити загальну онтологію компанії, що описує модель знань предметної області, яка потім застосовується в редакторі мережі для опису конфігурації бізнесу. Онтологія містить базові знання і відносини між ними, що подаються у формі семантичної мережі.

Моделююча система – програмний модуль, що дозволяє здійснювати моделювання ситуації за принципом « Що, якщо?..».

Еволюційний дизайн – модуль, що виробляє пропозиції щодо поліпшення конфігурації мережі в частині збільшення або зменшення певного числа ресурсів, зміни географії ресурсів і т.д.

Наведемо кілька прикладів побудови МАС для вирішення конкретних управлінських завдань.

2.4 МАС ДЛЯ РОЗПОДІЛУ ЗАМОВЛЕНЬ ТАКСІ

Для однієї з найбільших в світі компаній корпоративного таксі Addison Lee (Лондон)

було розроблено систему, що дозволила розподіляти і планувати приблизно 13 тисяч замовлень в день при наявності декількох тисяч власних машин (з них до 800 постійно на лінії), оснащених засобами GPS-навігації. При появі нового замовлення система автоматично знаходить найкращу машину, отримуючи відомості про координати найближчих машин на електронній карті Лондона, і попередньо бронює замовлення. Якщо ця машина вже зайнята, то починається ланцюжок переговорів, спрямований на вирішення виниклого конфлікту і досягнення компромісу, що дозволить перекинути попереднє замовлення на іншу машину, якщо це вигідно для всіх. Але і після цього робота системи з новим замовленням не зупиняється.

В середньому на подачу машини потрібно близько 15 хвилин, при цьому приблизно половину цього часу система продовжує безперервно шукати можливості для поліпшення перевезення з урахуванням замовлень, коли необхідно відправляти машину з урахуванням часу шляху проїзду до замовлення. Коли вже пора відправляти автомобіль на замовлення, система приймає остаточне рішення, посилає водієві повідомлення про параметри замовлення і чекає підтвердження про прийом замовлення.

Впровадження такої системи дало можливість зростання компанії за рахунок підвищення керованості. Істотно збільшилася ефективність автопарку (10 – 15%) завдяки оптимальному розподілу замовлень за рахунок мінімізації «холостого» пробігу і часу простою автомобілів. Також скоротилася кількість запізнень і час обслуговування заявок.

2.5 МАС ДЛЯ УПРАВЛІННЯ ГРУПАМИ ІНТЕЛЕКТУАЛЬНИХ РОБОТІВ

Хорошим прикладом інтелектуальних агентів служать роботи. Роботи можуть мати широкий асортимент штучних органів почуттів (сенсорні датчики) і штучних ефекторів (маніпулятори, педіпулятори). Їхня мобільність досягається завдяки колісним, гусеничним, крокуючим і іншим системам переміщення. Активність і автономність роботів тісно пов'язані з існуванням засобів визначення цілей і планування дій, систем підтримки вирішення завдань, а інтелектуалізація, крім володіння системою обробки знань, передбачає розвинені засоби комунікації різних рівнів, аж навіть до засобів природномовного спілкування.

Невід'ємним атрибутом інтелектуальних роботів є існування спеціальної підсистеми планування дій робота в реальних умовах навколишнього середовища, які визначаються рецепторами робота. Для планування діяльності робот повинен мати знання про властивості навколишнього середовища і шляхи досягнення цілей в цьому середовищі.

Розглянемо два підходи до управління групою інтелектуальних роботів.

Перший заснований на ринкових відносинах в групі роботів. Його суть полягає в наступному. Спочатку кожний робот генерує список цілей. Цілі, що відповідні відомим областям середовища, виключаються зі списків, а ті, що залишилися, розміщуються в порядку проходження. Кожний робот, по черзі, намагається продати кожне з цільових завдань всім іншим роботам групи, з якими можливий зв'язок, виставляючи їх на

аукціон. Кожен з решти роботів пропонує ціну, яка визначається приблизними витратами і доходами. Робот-аукціоніст, що пропонує цільове завдання, чекає поки всі роботи не запропонують ціну. Причому на це відводиться певний час. Якщо якісь роботи пропонують ціну, більшу за мінімум, запропоновану роботом-аукціоністом, то цільове завдання передається тому з них, який пропонує найбільшу ціну.

Після того, як всі аукціони проведені, роботи починають рух до першої своєї цілі. При її досягненні кожен робот генерує нові цілі і починає рух до наступної цілі, що міститься в завданні, пропонуючи всю решту в списку цільових завдань іншим роботам групи через аукціон. Однак в цьому підході використовуються дуже громіздкі механізми аукціонів і регулювання цін, що істотно обмежує його застосування.

Інший підхід до організації мультиагентної взаємодії в групах інтелектуальних роботів заснований на принципах колективного управління, характерних для колективів людей.

Колективом називають групу роботів, які вирішують загальне цільове завдання і взаємодіють між собою для вирішення цього завдання найкращим для групи чином. Колективна взаємодія – це взаємодія, що охоплює велику кількість елементів деякої системи і проявляється в їх узгоджених діях.

Метод колективного управління полягає в тому, що кожен робот групи: по-перше, самостійно керує процесом свого функціонування, тобто визначає свої дії, а по-друге, – погоджує ці дії з діями інших роботів групи, для того, щоб найбільш ефективно, тобто з мінімальними витратами і максимальною вигодою для групи, вирішити цільове завдання.

Основні принципи колективного управління:

- кожний член колективу групи самостійно формує своє керування (визначає свої дії) в поточній ситуації;
- формування управлінь (вибір дій) кожним членом колективу здійснюється тільки на основі інформації про колективну мету, що стоїть перед групою, ситуації в середовищі в попередній відрізок і в поточний момент часу, свій поточний стан і поточні дії інших членів колективу;
- в якості оптимального управління (дії) кожного члена колективу в поточній ситуації розуміється таке управління (дія), що вносить максимально можливий внесок у досягнення спільної (колективної) мети, або, іншими словами, дає максимально можливе збільшення цільового функціоналу при переході системи «колектив – середовище» з поточного стану в кінцеве;
- оптимальне управління реалізується членами колективу протягом найближчого відрізка часу в майбутньому, а потім визначається нове управління;
- допускається прийняття компромісних рішень, що задовольняють усіх членів колективу, тобто кожен член колективу може відмовлятися від дій, що приносять йому максимальну вигоду, якщо ці дії приносять малі вигоди або навіть шкоду колективу в цілому.

На відміну від групового управління, яке може бути як централізованим, так і децентра-

лізованим, колективне управління групою роботів завжди децентралізоване за своєю суттю. Тому описаний метод колективного управління роботами найбільш ефективний при реалізації в розподілених мультиагентних системах. Основна перевага – відносно низька обчислювальна складність алгоритмів, що дозволяє швидко приймати якщо не оптимальні, то близькі до них рішення в умовах динамічно змінюваної ситуації. Цей підхід застосовується не тільки для вирішення завдань колективного управління інтелектуальними роботами, але і для вирішення проблеми організації стійкого до відмов обчислювального процесу в розподілених багатопроцесорних інформаційно-керуючих системах складних динамічних об'єктів.

2.6 МАС ДЛЯ УПРАВЛІННЯ ГРУПОЮ БПЛА

Ще одним перспективним напрямком застосування МАС є системи безпілотних літальних апаратів. У сучасному світі безпілотні літальні апарати (англ. Unmanned Aerial Vehicles, далі БПЛА) набувають все більшої популярності в якості легких і недорогих інструментів для досліджень територій, розвідки і повітряних зйомок. В основній масі сучасна безпілотна техніка не має автономності (зазвичай дистанційно управляється оператором зі спеціального пульта). Є успішні розробки одиночних БПЛА, які під управлінням автопілота здатні в автономному режимі облетіти територію по заданому маршруту, збираючи ту чи іншу інформацію або виконуючи інші завдання. Аналіз більшості завдань, що вирішуються з використанням одиночних апаратів, показує, що вони можуть більш ефективно вирішуватися групою, оскільки у групи легких і маленьких літаків, що ефективно взаємодіють, з'являються нові корисні властивості.

ЛЕКЦІЯ 3. ПРОЕКТ «МУЛЬТИАГЕНТНА СИСТЕМА ДЛЯ БПЛА» ОПИС ТРИРІВНЕВОЇ СИСТЕМИ УПРАВЛІННЯ БПЛА ДЛЯ РЕАЛІЗАЦІЇ МУЛЬТИАГЕНТНОЇ ВЗАЄМОДІЇ.

3.1 ВСТУП

В цей час мініатюризація електронних компонентів та технологічність їх виготовлення дозволяє застосовувати невеликі і доступні безпілотні літальні апарати (БПЛА) для виконання завдань дослідження території.

Наприклад, при використанні групи БПЛА для геологорозвідки на певній території робота системи організовується так:

Обирається тип завдання (в обраному прикладі – геологорозвідка).

Залежно від площі досліджуваної території і кількості БПЛА в групі, їхніх характеристик, територія поділяється на ділянки і формуються окремі завдання для кожного члена групи.

У мікрокомп'ютер кожного БПЛА групи записується глобальне завдання (параметри досліджуваної території і т.ін.) і окрема задача цього літака-агента.

Кожен агент приступає до виконання поставленого йому завдання.

Коли в зону Wi-Fi видимості одного БПЛА з групи потрапляє інший, то при «спілкуванні» відбувається передача між агентами накопиченої інформації і при необхідності взаємне уточнення окремих завдань. Таким чином, під час виконання приватного завдання всі агенти накопичують інформацію про хід вирішення загального завдання групи, а також локально приймають рішення про коригування своїх приватних завдань для більш ефективного виконання загальної задачі. Наприклад, група БПЛА летить за певним маршрутом, всі БПЛА летять на різній висоті, при обміні інформацією між членами групи під час виконання завдання вся група оперативно перебудовується на цю висоту.

Базові наземні станції, забезпечуючи зв'язок із центром обробки даних (ЦОД), приймають/передають інформацію від БПЛА, що перебувають у зоні їхньої видимості або підтримують зв'язок. Оскільки в процесі спілкування між БПЛА інформація про виконання загального завдання накопичується у всіх комп'ютерах, то дані навіть від тих літаків, які рідко виходять на зв'язок, все одно потрапляють в ЦОД.

Отримана в ЦОД інформація обробляється і візуалізується для замовника (видається карта з нанесеними досліджуваними характеристиками).

Наявність зворотного зв'язку з мобільними агентами (БПЛА) дозволяє оперативно формувати інструкції з ЦОДу щодо коригування їх завдань.

У світі IT вже у багатьох областях переважає схема, в якій немає чіткого виділення трьох ізольованих стадій: підготовка даних, розрахунок, аналіз результатів. Цьому сприяло загальне проникнення персональних комп'ютерів, смартфонів, мобільних телефонів,

які кардинально змінили уявлення про місце ЕОМ в звичайному житті. Комп'ютери вже інтегровані у багато сфер діяльності, розробники вбудовують мініатюрні обчислювальні пристрої і спеціалізовані програми у середину багатьох процесів. У масовому сприйнятті про роль ЕОМ в практичному житті вже сталося зрушення від суперрозрахунків до мобільності, яка розуміється в дуже широкому сенсі.

Виходячи із наведеного вище, перспективну модель обчислення можна представити таким чином: мобільні пристрої, що мають різні датчики і сенсори (наприклад, відео- або фотокамеру, gps, датчики температури і тиску і т. ін.) і здатні «спілкуватися» одне з одним, є збирачами інформації, вони передають інформацію на суперкомп'ютер. Мобільні пристрої, збираючи інформацію, одночасно можуть «фільтрувати» її, тобто визначати особливості і розділяти дані, різні за тематикою (наприклад, відокремити файли з координатами і файли з фотографіями).

У цьому розділі буде розглянута трирівнева система керування БПЛА, на основі якої будується мультиагентна система для керування легкими БПЛА.

3.2 МУЛЬТИАГЕНТНИЙ ПІДХІД І ТРИРІВНЕВА СИСТЕМА КЕРУВАННЯ БПЛА

Комплекс БПЛА, як правило, складається з одного БПЛА і базової станції під керуванням людини. Людина з базової станції визначає завдання для БПЛА, останній, в свою чергу, виконує його і передає інформацію на базову станцію, при цьому людина здатна відстежувати хід виконання завдання. Іноді використовують кілька комплексів, при цьому кожний БПЛА прив'язаний до своєї базової станції. Обмін інформацією між комплексами відбувається тільки на землі, між базовими станціями [18]. Таку роботу комплексу можна представити, як дворівневу систему керування. Верхній рівень управління здійснюється базовою станцією. Тут визначається глобальне завдання для БПЛА, задається висота польоту, швидкість польоту, маршрут, точки збору інформації і т. д. Нижній рівень управління здійснюється автопілотом БПЛА, який діє за записаною на ньому програмою і спілкується з базовою станцією. Для виконання завдання автопілот управляє виконавчими механізмами, збирає інформацію з датчиків і відправляє дані на базову станцію.

Описана вище система управління комплексом і принцип застосування декількох комплексів БПЛА для групового виконання завдання характеризуються відсутністю автономної постановки нових завдань, що дозволяє групі оперативно приймати ефективні рішення щодо зміни сценарію виконання поставленого завдання. Типовими прикладами подій, що викликають необхідність у постановці нових завдань, є такі: поява нової корисної інформації для більш ефективного виконання завдання; вихід з ладу частини наявних ресурсів; зміна критеріїв прийняття рішень. Чим вище невизначеність, чим більш розподілений характер мають процеси прийняття рішення і чим частіше трапляються незаплановані події, тим нижче ефективність існуючих систем, нездатних самостійно приймати рішення і автоматично перебудовуватися під зміни в середовищі. Крім того, це робить розробку і експлуатацію розглянутих систем вкрай дорогими.

Для вирішення подібних проблем застосовуються мультиагентні технології. В основі цих технологій лежить поняття «агента» – програмного об'єкта, здатного сприймати ситуацію, приймати рішення і взаємодіяти з подібними собі. Характерними особливостями інтелектуальних агентів є такі:

- колегіальність, тобто здатність до колективної цілеспрямованої поведінки в інтересах вирішення загального завдання;
- автономність, тобто здатність самостійно вирішувати локальні завдання;
- активність, тобто здатність до активних дій задля досягнення спільних і локальних цілей;
- інформаційна і рухова мобільність, тобто здатність активно переміщатися і цілеспрямовано шукати і знаходити інформацію, енергію і об'єкти, необхідні для колективного вирішення спільного завдання;
- адаптивність, тобто здатність автоматично пристосовуватися до невизначених умов в динамічному середовищі.

Ці можливості кардинально відрізняють мультиагентні системи (МАС) від існуючих «жорстко» організованих систем управління групи автономних БПЛА.

Ми застосовуємо мультиагентний підхід для управління групою БПЛА, заснований на «спілкуванні» моделей одне з одним, при цьому утворюється нове джерело інформації для коригування польоту автопілотом.

Бажано виконання двох основних вимог для розробки мультиагентної системи для групи БПЛА. По-перше, на кожній моделі має бути невеликий, але потужний мікрокомп'ютер, з можливістю автономної зміни завдання для автопілота і спілкування в мультиагентній системі. По-друге, необхідно організувати стійкий зв'язок між агентами групи.

Визначимо основні корисні властивості, які має група взаємодіючих моделей, у порівнянні з використанням одного БПЛА:

- колективний обліт створює велику об'ємну картину світу;
- взаємодія між собою допомагає оптимізувати маршрут польоту, грунтуючись на вже наявних даних з іншого БПЛА;
- більше множин стратегій;
- можливість створення об'єктів;
- більш ефективне вирішення завдань (екологія, метеорологія, оптимізація польоту групи);
- більше гарантій виконання завдання;
- виграш у часі (головна умова для задач пошуку загублених людей);
- можливість одночасного обстеження території;
- можливість коригування плану і вибору оптимального маршруту, грунтуючи дії на наявних даних сусіднього БПЛА;
- можливість постановки різних завдань для різних учасників групи БПЛА.

Для застосування мультиагентного підходу до групи БПЛА в дворівневу схему управління комплексом БПЛА ми додаємо проміжний середній рівень, який створюється за

рахунок додаткового мікрокомп'ютера.

Таким чином, ми отримуємо нову трирівневу систему управління БПЛА-агента (рис. 3.1), В ній на верхньому рівні відбувається формування глобального завдання вже для групи БПЛА, початкові умови, коригування або зміни глобального завдання, а також прийом і обробка отриманої від групи інформації. На верхньому рівні знаходиться персональний комп'ютер, до якого приєднані різні модулі для зв'язку (модеми, радіоприймачі та ін.). Середній рівень управління здійснюється за рахунок впровадження в БПЛА мікрокомп'ютера, який здатний обмінюватися інформацією з іншими членами групи і базовою станцією. На основі цієї інформації мікрокомп'ютер, якщо це буде потрібно, може змінити своє початкове завдання для прискорення вирішення глобального завдання. Таким чином, ми отримуємо групу БПЛА-агентів, яка здатна адаптуватися до змін. На нижньому рівні знаходиться управління виконавчими механізмами, датчиками і додатковим обладнанням.

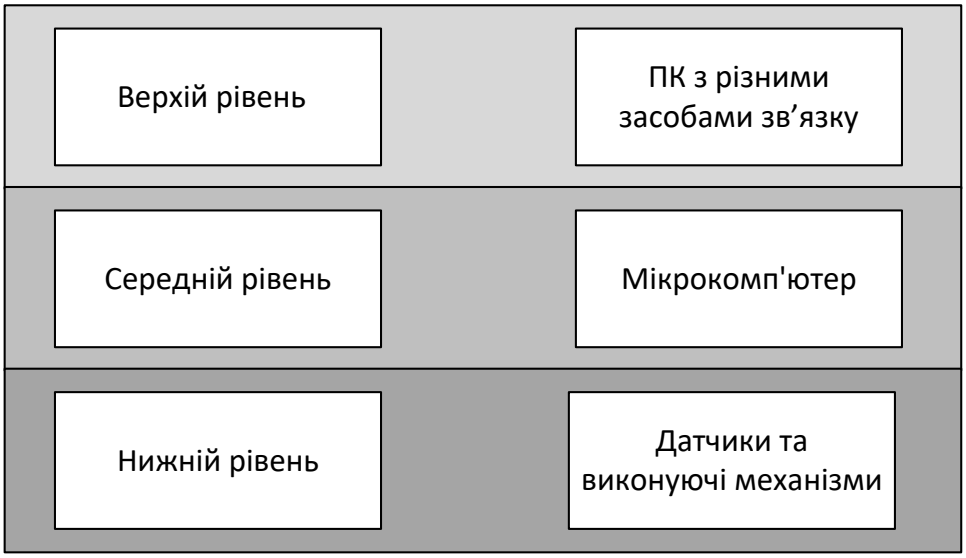

Рис. 3.1. Три рівня керування БПЛА-агента

3.3 БПЛА-АГЕНТ ДЛЯ АВТОНОМНОЇ ГРУПИ

Вище було описано систему управління групою БПЛА. Основна відмінність групи від звичного комплексу БПЛА – це автономне спілкування БПЛА-агентів між собою. Для того, щоб реалізувати мультиагентну систему для управління групою БПЛА, необхідно переглянути апаратне оснащення одиночного БПЛА.

Розглянемо наведену в попередньому розділі трирівневу систему з точки зору апаратної реалізації.

На верхньому рівні управління нами застосовується комп'ютер – базова станція (ноутбук, нетбук або стаціонарний ПК), оснащений різними засобами зв'язку (радіомодемом, Wi-Fi модемом, Інтернет модемом) (рис. 3.2). Основними завданнями базової станції є такі:

- визначення глобальної місії групи БПЛА-агентів (параметри території дослідження, завдання способів дослідження, висота польоту і т. д.);
- розбиття глобальної задачі на частини і формування початкового індивідуального

завдання для кожного БПЛА-агента групи в залежності від кількості БПЛА-агентів і поставленого перед групою завдання;
- обмін інформацією з БПЛА-агентами, що знаходяться в зоні зв'язку;
- збір та обробка інформації від БПЛА-агентів групи;
- формування нової глобальної місії для групи в залежності від тієї нової інформації, що надходить на базову станцію.

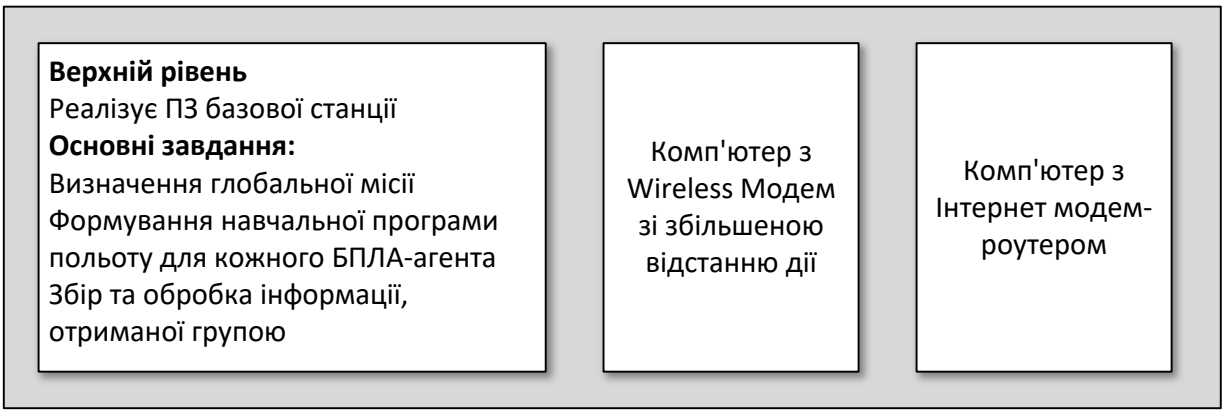

Рис. 3.2. Верхній рівень управління БПЛА-агента

На середньому рівні управління БПЛА-агента знаходиться бортовий мікрокомп'ютер – головний пристрій системи управління БПЛА. Його основна мета – виконати поставлене завдання з найменшими витратами часу і ресурсів. Для цього він виконує п'ять основних функцій, а саме:
- генерація оновлень до програми польоту для автопілота;
- обробка даних навігаційного обладнання та телеметрії;
- робота з додатковим обладнанням;
- спілкування з мікрокомп'ютерами інших БПЛА-агентів (якщо робота ведеться в групі);
- передача даних на базову станцію і отримання від неї нових завдань.

На базовій станції мікрокомп'ютер отримує загальні завдання (початковий стан, кінцеві точки, завдання для групи і т. ін.). При груповому польоті спільне завдання розбивається на окремі для кожного БПЛА. У процесі виконання завдання мікрокомп'ютер спілкується з іншими членами групи, що знаходяться в радіусі дії радіозв'язку. Взаємодія в групі дозволяє групі БПЛА ефективніше виконати спільне завдання, а також уникнути зіткнень.

На основі даних, отриманих від базової станції, з навігаційного обладнання і при спілкуванні з іншими БПЛА, мікрокомп'ютер може створити нову програму для автопілота, якщо стара не підтримує необхідних вимог для виконання спільного завдання.

При прольоті БПЛА в зоні зв'язку з базовою станцією мікрокомп'ютер відправляє накопичені дані і отримує нові завдання. При цьому накопичення даних відбувається не тільки за рахунок власних сенсорів і датчиків, а й при зв'язку з іншими БПЛА. Як говорилося вище, у мікрокомп'ютер закладається окреме для нього завдання і завдання для групи. У процесі спілкування, мікрокомп'ютер накопичує інформацію про виконання завдання групою. Дані про виконання свого окремого завдання він отримує

сам, використовуючи бортові датчики.

Зв'язок з базовою станцією відбувається по окремому радіоканалу або через GPRS по GSM модему. GSM модем легко інтегрується з мікрокомп'ютером, але для передачі даних пакети необхідно стискати.

Зв'язок між мікрокомп'ютерами різних БПЛА здійснюється за рахунок вбудованого радіоприймача з частотою у 2,4 GHz і протоколом спілкування 802.11 n (Wi-Fi), в якому застосовується технологія, що зв'язує два найближчі канали в один. Таким чином, мікрокомп'ютери в БПЛА можуть одночасно приймати і відсилати інформацію один до одного. Зв'язок з базовою станцією відбувається по окремому радіоканалу або через GPRS по GSM модему.

За рахунок невеликої ваги, БПЛА зліт здійснюється з рук людини або з катапульти. Посадка – або за рахунок вбудованого парашута, або за рахунок «перехоплення» на ручне керування.

Нижній рівень управління здійснюється автопілотом БПЛА-агента (рис. 3.3). Автопілот – пристрій з мікроконтролером, з системою реального часу. Основне завдання автопілота полягає в керуванні виконавчими механізмами (сервоприводами, мотоустановками, додатковим обладнанням), грунтуючись на записаній для нього програмі польоту і інформації з датчиків (інерціальної системи, інфрачервоних датчиків, датчиків тиску і швидкості і т. ін.).

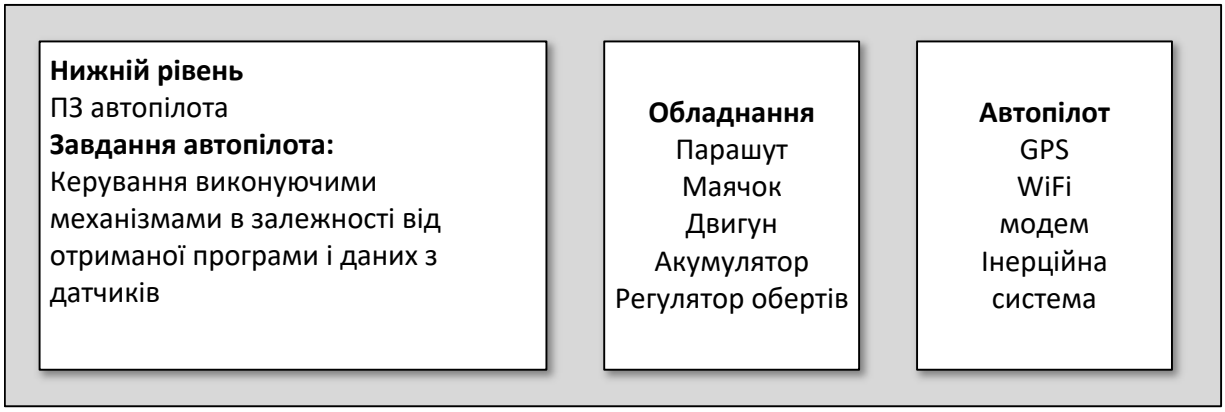

Рис. 3.3. Нижній рівень керування БПЛА-агентом

Таким чином, апаратна реалізація трирівневої системи, яка потрібна для створення мультиагентної мережі управління групою БПЛА, є можливою.

3.4 АЛГОРИТМИ ГРУПОВОЇ РОБОТИ МЕРЕЖІ БПЛА

У цьому розділі ми розглянемо два основні типи алгоритмів управління польотом БПЛА: алгоритм моніторингу місцевості на прикладі дослідження екологічної ситуації та алгоритм оптимізації польоту БПЛА на прикладі використання термічних потоків для збільшення дальності польоту.

При використанні групи БПЛА для моніторингу місцевості (розглянемо задачу дослідження екологічної обстановки в акваторії затоки, – наприклад, розлив нафти) робота

системи організовується так:

- Вибирається тип завдання (в обраному прикладі – пошук нафтових плям і джерела їх утворення).
- Залежно від площі досліджуваної території, кількості БПЛА в групі та їхніх характеристик територія поділяється на ділянки, і формуються окремі завдання для кожного члена групи (для обраного прикладу завдання буде стояти таким чином: пошук аномалії інтенсивності кольору поверхні акваторії).
- У мікрокомп'ютер кожного БПЛА записується глобальне завдання (параметри досліджуваної території і т. ін.) і окреме завдання для цього БПЛА – агента.
- Кожен агент приступає до виконання поставленого йому завдання.

Коли в зону Wi-Fi видимості одного БПЛА з групи потрапляє інший, при «спілкуванні» відбувається передача між агентами накопиченої інформації і, при необхідності, взаємне уточнення окремих завдань. Таким чином, під час виконання окремого завдання всі агенти накопичують інформацію про хід вирішення загального завдання групи, а також локально приймають рішення про коригування своїх приватних завдань для більш ефективного виконання загального. (Наприклад, для обраного нами прикладу: група БПЛА досліджує територію на предмет нафтових плям, усі БПЛА досліджують різні квадранти. При обміні інформацією між членами групи виявляється, що є нафтові плями в одному з квадрантів. По ходу виконання завдання вся група перебудовує завдання, яке зводиться до пошуку джерела розливу нафти).

Базові наземні станції, забезпечуючи зв'язок з ЦОД, приймають/передають інформацію від БПЛА, які перебувають у зоні їхньої видимості або підтримують зв'язок через інтернет. Оскільки в процесі спілкування між БПЛА інформація про виконання загального завдання накопичується у всіх комп'ютерах групи, то дані навіть від тих літаків, які рідко виходять на зв'язок, все одно потрапляють в ЦОД.

Отримана інформація в ЦОДі обробляється і візуалізується для замовника (видається карта з нанесеними досліджуваними характеристиками).

Наявність зворотного зв'язку з мобільними агентами (БПЛА) дозволяє оперативно формувати з ЦОДу інструкції щодо коригування їх завдань.

Такий алгоритм дії групи підходить до будь-яких завдань з візуального моніторингу місцевості. Основні відмінності будуть складатися в типі шуканого сигналу і в відповідному додатковому устаткуванні для пошуку цього сигналу.

3.5 ПРОГРАМУВАННЯ БОРТОВИХ МІКРОКОМП'ЮТЕРІВ

Вище було показано переваги групової взаємодії і принцип побудови системи управління БПЛА, заснований на трирівневій архітектурі. Така архітектура системи управління БПЛА характеризується поділом конкретних «обов'язків» між обчислювальними блоками. Розглянемо логіку і принципи розробки програм для трирівневої архітектури.

При роботі трирівнева система формує керуючі дії на основі вхідних даних. Для коректного виконання завдання при створенні керуючої дії в програмі автопілота за прин-

ципом різних контурів управління формуються три послідовні набори даних, що забезпечують:

- підтримку заданих параметрів руху (керуючі дії на виконавчі механізми, які генеруються автопілотом на підставі даних, отриманих від мікрокомп'ютера);
- коригування курсу (дані для управління передаються автопілоту і формуються на основі даних навігаційної системи і поставленого завдання для мобільного об'єкта);
- виконання маршруту руху (формуються на базовій станції в залежності від інформації).

Всі три контури управління в кінцевому підсумку формують керуючий сигнал на виконавчі механізми.

Схематично три контури показані на рис. 3.4.

Перший контур – створення керуючого сигналу на основі інформації. Автопілот відстежує відхилення мобільного об'єкту на підставі інформації від інерційних датчиків (таких, як інфрачервоні датчики, магнітометри і гіроскопи) і коригує кути крену, швидкість, висоту і т.ін. для підтримки заданих параметрів руху мобільного об'єкту.

Автопілот формує керуючу дію на сервомеханізми із застосуванням пропорційно-інтегрально-диференціального (ПІД) регулятора. Це найбільш часто використовуваний вид управління зі зворотним зв'язком. Якщо $u(t)$ – керуючий вплив, $y(t)$ – результати вимірювання, $r(t)$ - бажаний результат вимірювання, $e(t) = r(t) - y(t)$ – «неув'язка», то ПІД регулятор має наступний вигляд:

$$u(t) = K_P e(t) - K_I \int e(t)dt + K_D e(t)$$

де K_P, K_I, K_D – коефіцієнти посилення пропорційної, інтегральної і диференціальної складової регулятора. У стійких системах стабілізацію руху часто може бути забезпечено тільки за допомогою пропорційної складової. Інтегральний член компенсує статистичну помилку, а диференціальний служить для компенсації затримок у системі. Основою для створення керуючого сигналу є дані, одержувані з різних датчиків і джерел отримання додаткової інформації (наприклад, інформація з базової станції, отримана через модуль зв'язку, і т. ін.).

Другий контур необхідний для формування поправки на відхилення від заданого напрямку руху і коригування курсу. Одна з істотних причин відхилень легких мобільних об'єктів – зміни напрямку вітру або нерівності поверхні землі. Також на відхилення впливають випадкові і систематичні помилки в одержуваних в процесі польоту даних: помилки в навігаційних системах, дискретність в отриманні даних і т. ін. Для боротьби з цією проблемою застосовують фільтри, за допомогою яких проводиться пророкування зміщення в наступний момент часу і внесення на основі цих даних поправок в курс БПЛА.

Для того, щоб кожного разу не заходити в програмний код, в автопілоті передбачено зміну плану польоту в процесі його виконання. Як правило, це доступний для коригування текстовий файл, в якому зберігаються дані про координати.

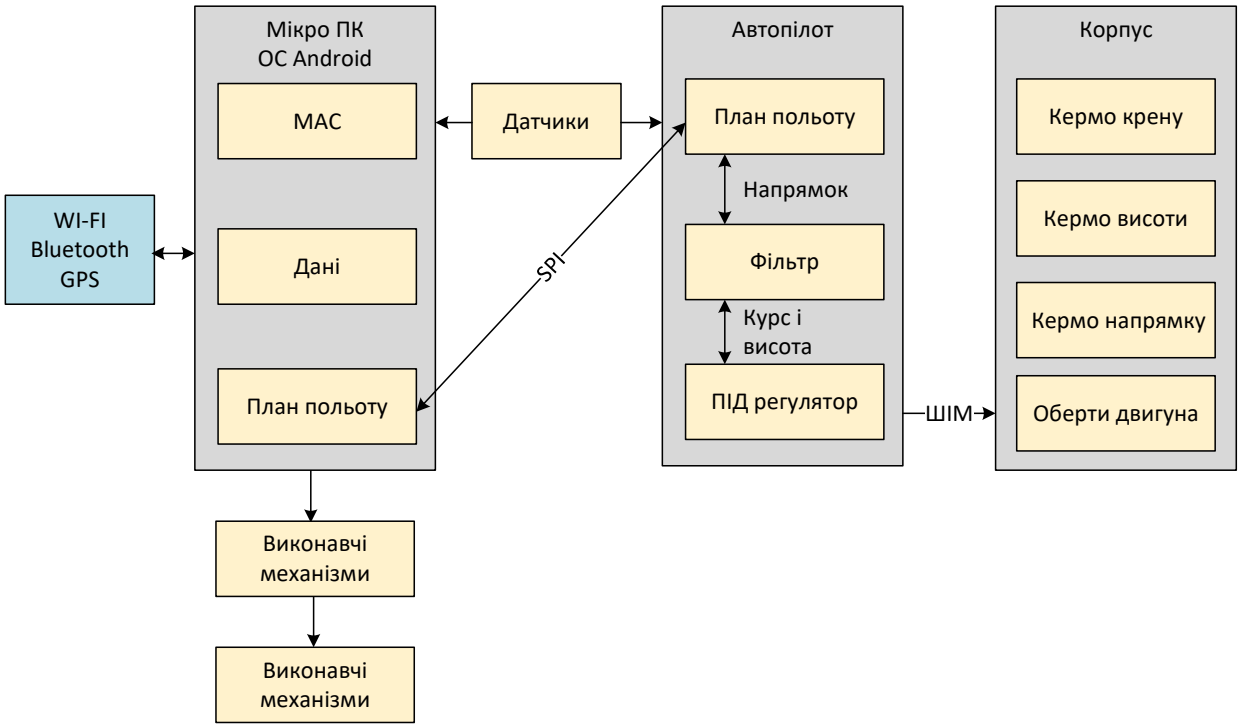

Рис. 3.4. Три контури управління

Обмін інформацією між мікрокомп'ютером і автопілотом можна здійснювати за допомогою послідовного периферійного інтерфейсу (SPI, Serial Peripheral Interface).

У третьому контурі управління на основі поставленого глобального завдання для групи мобільних об'єктів і окремого завдання для агента обираються точки маршрутів. Спочатку точки маршруту формуються оператором на базовій станції на основі поставленого завдання. Глобальне завдання записується в мікрокомп'ютер, а початкові приватні завдання для членів групи – в їхній автопілот. У процесі польоту зміни глобального завдання надсилаються мікрокомп'ютеру, який ці зміни обробляє і формує нові приватні завдання для автопілотів. Також, виходячи з прийнятої інформації від інших мобільних об'єктів, в рамках глобального завдання для більш швидкого його завершення, мікрокомп'ютер може змінити приватне завдання для автопілота. В описаному вище прикладі трирівневої архітектури обмін інформацією між мобільними об'єктами і між мобільним об'єктом і базовою станцією відбувається за допомогою: радіо-, Wi-Fi-, інтернет-модему. Зв'язок за допомогою GSM модему з виходом у всесвітню систему інтернет в основному передбачений для передачі необхідної інформації на сервер, доступ до якого можна здійснити з будь-якого комп'ютера, підключеного до «всесвітньої павутини».

Конкретизуємо алгоритм (протокол) взаємодії у групі мобільних об'єктів без єдиного центру керування, але з єдиним центром збору даних моніторингу деякої території за допомогою різного обладнання для таких завдань, як:

- візуальний пошук об'єктів;
- пошук джерела радіосигналу;
- пошук витоку газопродуктів;
- пошук джерела магнітного випромінювання;
- пошук джерела радіаційного випромінювання і т. ін.

Кожному комплекту даних (фотографії, набору даних по виміру хімічного складу і т. ін.) приписується свій індекс, який відповідає часу фіксації і координатам точки, в якій проведений збір даних.

Якщо встановився зв'язок, то відбувається наступне:
- відправка набору індексів наявної інформації;
- прийом набору індексів;
- порівняння набору прийнятих індексів з наявним набором індексів;
- виділення нових індексів з прийнятого набору;
- відправлення запиту на отримання нових даних з новими індексами;
- прийняття запиту на відправку індексів;
- формування пакету даних на основі запитаних індексів;
- передача даних;
- прийняття нових даних, запис в пам'ять.

На основі нових даних мікрокомп'ютером може бути прийнято рішення про зміну програми для автопілота для коригування району пошуку. .

Застосування групи мобільних об'єктів підвищує гарантії пошуку. Наприклад, можлива ситуація, при якій один з мобільних агентів виходить з ладу. Це може бути різке падіння одного з мобільних об'єктів. Для завершення виконання глобальної місії, для групи виникає задача змінити маршрути інших об'єктів. При цьому алгоритм роботи мобільного об'єкта такий:
- завершити свою задачу;
- порівняти наявні дані з усією картою необхідних даних в глобальному завданні;
- виділити область даних сусідніх ділянок;
- визначити незаповнені дані в сусідніх ділянках;
- якщо є незаповнені дані, то сформувати новий маршрут для автопілота;
- якщо всі дані заповнені, відправити повідомлення про завершення завдання.

При виконанні такого алгоритму мобільний об'єкт визначає, чи потрібно допомогти сусідам. Якщо так, то розширює поле своєї роботи.

При відправці даних на базову станцію, необхідність зберігати ці дані в мікрокомп'ютері відпадає, і в базі даних ставиться позначка про час відправки цієї інформації на сервер. Далі, при необхідності звільнення місця, вона стирається з пам'яті. При моніторингу території дані з близькими координатами відзначаються як повторні, але з різним часом отримання, і порівнюються на сервері з метою пошуку змін.

Обмін інформацією мобільного об'єкта з базовою станцією відбувається, коли він входить у зону видимості базової станції. Такий зв'язок здійснюється за рахунок стабільного радіоканалу з радіусом дії 1 – 2 км. На більшій відстані мобільний об'єкт спілкується через GSM. Передача інформації по такому каналу може бути перервана через збої зв'язку при використанні у повітрі і на значній відстані від антен GSM. У зв'язку з цим алгоритм відправки пакетів такий:
- створити пакет з отриманими даними в установленому інтервалі часу (наприклад, фотозображення, зроблені за першу хвилину збору даних);

- стиснути пакет;
- якщо немає пакетів в черзі відправки, то відправити стислий пакет;
- якщо є черга відправки, то поставити пакет в чергу відправки;
- після відправки пакету і отримання підтвердження доставки поставити позначку в індексах – «відправлено на сервер». В іншому випадку пакет вважається невідправленим.

У ЦОДі дані обробляються для вилучення необхідної інформації. Для обробки фотографій застосовуються програмні додатки з алгоритмами розпізнавання образів, виділяються фотографії з однаковою місцевістю і на них визначаються зміни. На виході споживач отримує фотографії із зазначеною найбільш важливою інформацією. Одним з результатів обробки даних може стати формування нової глобальної місії для групи мобільних об'єктів.

Лабораторні роботи

ЛАБОРАТОРНА РОБОТА № 1.
УСТАНОВКА ПЗ ДЛЯ РОЗРОБКИ І ТЕСТУВАННЯ ДОДАТКІВ

Основні прийоми розробки додатків для ОС Android, процес установки і настройки середовища розробки, знайомство з Android Virtual Device і призначеним для користувача інтерфейсом (UI), проведення сеансу налагодження.

1.1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Освоїти основні прийоми розробки додатків для ОС Android, вивчити процес установки і настройки середовища розробки, знайомство з Android Virtual Device і призначеним для користувача інтерфейсом (UI), проведення сеансу налагодження.

Презентацію до лабораторної роботи Ви можете завантажити тут.

1.2 ІНСТРУКЦІЯ ПО ВИКОНАННЮ ЛАБОРАТОРНОЇ РОБОТИ

ПІДГОТОВКА

Перед тим, як приступати до створення додатків на Android, необхідно вибрати і встановити відповідне середовище розробки і встановити відповідний інструментарій для розробки додатків під ОС Android – Android SDK.

Як середовище розробки ми будемо використовувати Eclipse IDE. Всі приклади будуть наведені для цього середовища розробки. Звичайно, можна вибрати будь-які інші, наприклад, Intellij IDEA.

Переконайтеся в тому, що ваше ОС задовольняє системним вимогам для установки Eclipse IDE і Android SDK, описаним в http://developer.android.com/sdk/index.html.

Для початку розробки знадобиться набір з Eclipse IDE, Android SDK і JDK (Java Development Kit). Весь набір програм можна скачати на офіційному сайті Android для розробників за посиланням: http://developer.android.com/sdk/index.html. Також, програми доступні на офіційних сайтах за посиланням: http://www.oracle.com/technetwork/java/javase/downloads/index.html (JDK), http://www.eclipse.org/downloads/(Eclipse IDE). Звернемо увагу, що при скачуванні програм з різних джерел необхідно перевірити, чи сумісна версія Eclipse з Android SDK.

Android SDK являє собою засіб розробки додатків під ОС Android, і поширюється у вигляді ZIP або EXE файлів. Android SDK складається з пакетів, вміст яких можливо довантажувати з допомогою Android SDK Manager. В Android SDK входять наступні пакети:

- пакет з версіями платформ (SDK Platform) – містить папки з версіями платформ, в кожну з яких входить файл Android.jar – файл архіву Java, що містить всі класи SDK Android, необхідні для створення додатків;
- пакет з образами систем (System Image) – образ системи використовується в іммуляторі Android пристрою при налагодженні і розробці додатків;
- пакет з вихідним кодом платформи Android (sources);

- пакет документації (Documentation) – документація SDK надається локально і через Інтернет. В основному вона виконана у форматі Javadocs, що дозволяє легко орієнтуватися в безлічі пакетів SDK;
- каталоги інструментів (SDK Tools, SDK Platform – tools) – містять всі інструменти для налагодження і тестування Android – додатків;
- USB – driver – каталог, що містить всі необхідні драйвери для підключення до підтримуючих Android пристроїв.

Початковий пакет Android представляє тільки основні інструменти. Інші компоненти, при необхідності, можна завантажити та встановити (наприклад, GoogleAPI).

Android пропонує плагін ADT (Android Developer Tools) для Eclipse IDE, який забезпечує розробників ефективними інструментами для створення Android – додатків. Він розширює багаті можливості Eclipse новими корисними утилітами. Наприклад, після установки ADT на панелях toolbar і menubar з'являються «контролі», котрі дозволяють швидко створювати проект.

Для установки ADT, необхідно відкрити панель розширення середовища (InstallNewSoftware) в закладці «Help», а потім у вікні додати новий шлях https://dl – ssl.google.com/android/eclipse/.

Далі Eclipse виконає пошук необхідних даних, і у випадаючому списку ви зможете вибрати ADT і встановити його, натиснувши Install. Запускається процес інсталяції плагіна. Якщо в ході установки виникнуть помилки, то, можливо, що зазначений ресурс більше через якісь причини недоступний, або ви використовуєте стару версію Eclipse, або не встановили JDK.

Запустіть AVD Manager прямо з Eclipse (Window – > Android SDK and AVD Manager). На екрані, що міститься зліва, є список з трьох основних компонентів: віртуальні пристрої, встановлені і доступні пакети. У встановлених пакетах знаходяться всі компоненти, встановлені за замовчуванням або за потребою. За допомогою вкладки «доступні пакети» можна автоматично завантажувати і встановлювати необхідні рішення. Наприклад, ми можемо встановити Google API's, з метою використання в додатку Google (карти, пошук).

Перш ніж писати додаток з використанням Android API's необхідно створити віртуальний пристрій, котрий дозволить тестувати написане. Щоб створити віртуальний пристрій в головному вікні Android SDK та AVD Manager, клікніть на кнопку New, яка дозволяє налаштувати віртуальну машину. Заповнюйте поля зверху вниз. Спливаючі міні-віконця допоможуть вам при заповненні. На цьому установка Android повністю завершена, тепер можна приступати до написання додатків.

Кнопки є невід'ємною частиною будь-якого інтерактивного додатку. ОС Android володіє чудовими засобами створення та управління кнопками. Кнопка описана класом Android.widget.Button, який є суперкласом для всіляких типів кнопок (флажкова кнопка, радіокнопка, перемикач).

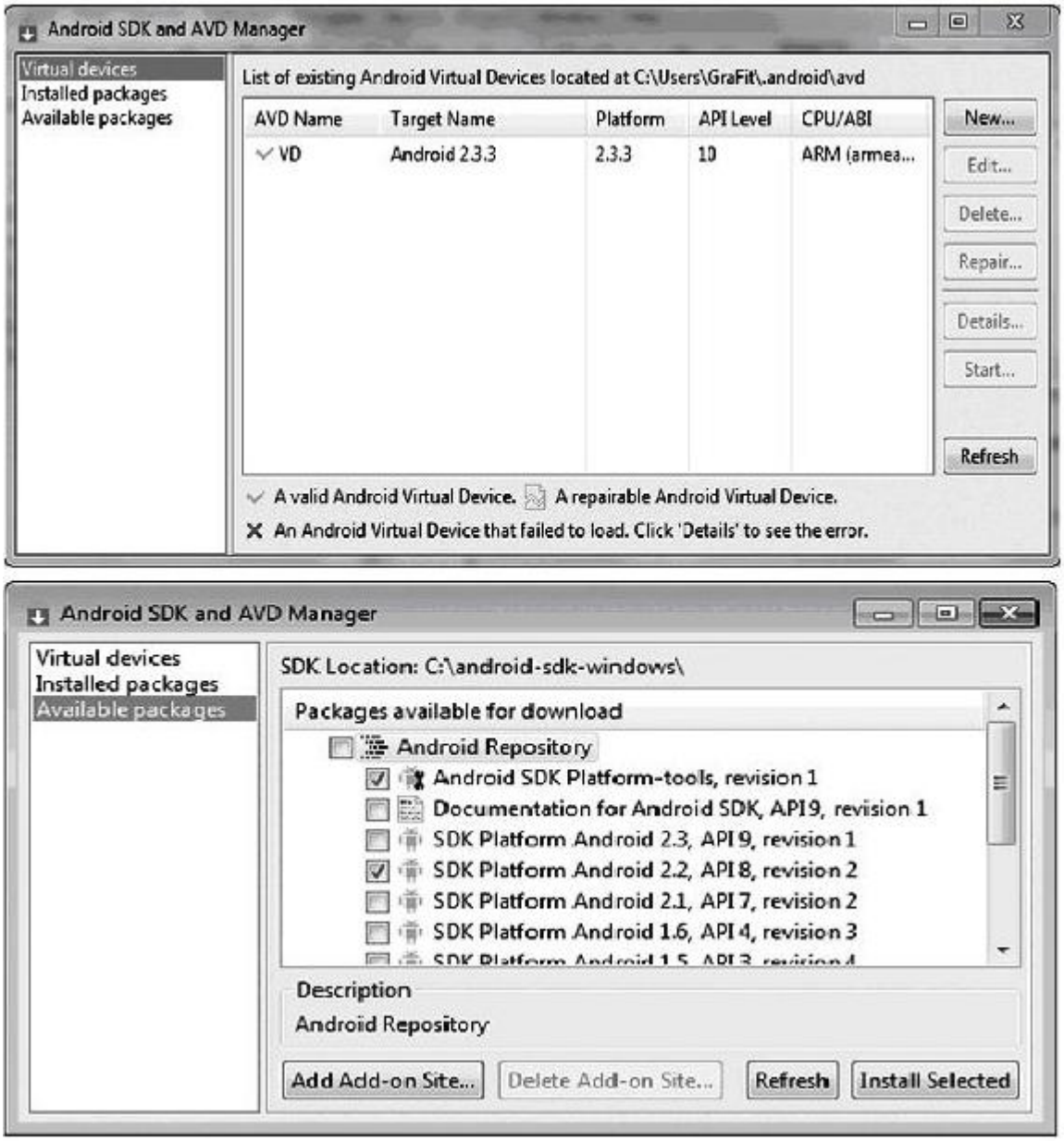

Рис. 1.1 Android API для елементів UI

У кнопки визначено кілька конструкторів:

- Button (Context context);
- Button (Context context, AttributeSetattr);
- Button (Context context, AttributeSetattr, intdefStyle).

Context – це клас який містить інформацію про середовище програми та надає доступ до різних системних ресурсів і класів.

Поля текстового введення. Android.widget.EditText є невеликим облицюванням навколо TextView, що дозволяє створювати редагований текст.

EditText має кілька кнструкторів:

- EditText (Contextcontext);
- EditText (Contextcontext, AttributeSetattr);
- EditText (Context context, AttributeSetattr, intdefStyle).

клас EditText містить кілька методів. Розглянемо де-які з них:

- public void selectAll (Spannable); метод фіксує цілком як текст вказаний ресурс. Spannable – це інтерфейс для організації тексту у вигляді блоків (шматків тексту).
- public Editable getText (); повертає текст, який відображається TextView. Editable інтерфейс для опису динамічного тексту, вміст і розмітка якого можуть бути змінені.
- public void setText (CharSequence, TextView.BufferType); встановлює текст, який відображається в TextView. Перший параметр визначає текст, другий – спосіб його зберігання: динамічний текст, нормальний текст (default), блоковий текст.

Списки Android.widget.ListView – клас, що відображає набір елементів у вигляді списку з можливістю прокрутки. Елементи, асоційовані з цим поданням, описуються класом ListAdapter.

ListAdapter – успадковує базовий клас Adapter і служить мостом між даними і ListView. Часто дані можуть бути представлені курсором, але необов'язково. Зручність у тому, що ListView може відображати будь-які дані, аби вони були загорнуті в ListAdapter. ListAdapter має кілька підкласів (ArrayAdapter, BaseAdapter), які призначені для різних цілей. Наведемо приклад використання ArrayAdapter.

ArrayAdapter – представляє дані у вигляді масиву і додає зручний функціонал (додавання, видалення, пошук) для роботи з ними.

- ArrayAdapter (Context, TextViewResourceId);
- ArrayAdapter (Context, TextViewResourceId, T [] objects).
- TextViewResourceId – посилання на xml – файл уявлення, даних списку. Параметр T [] objects задає відображені дані.

LinearLayout. Лінійний менеджер розміщення використовується для відображення елементів у вигляді лінійних списків, горизонтальних або вертикальних. Причому, можна використовувати кілька Layouts одночасно, в цьому випадку вони будуть ділити екран.

- LinearLayout (Context context);
- LinearLayout (Context context, AttributeSet);
- LinearLayout (Context context, AttributeSet, intdefStyle).

Розглянемо основні методи:

- generateLayoutParams (AttributeSet attr)- задає параметри layout (ширина, довжина, фон, тяжіння);
- getOrientation ()- задає напрямок розміщення.

RelativeLayout. Цей менеджер розміщення використовується в тому випадку, коли необхідно розміщувати елементи якогось віджета. Позиція розміщення задається у вигляді: елемент X розташувати нижче елемента Y.

RelativeLayout дуже потужний засіб розташування елементів, що не залежить від розмірів вікна програми чи конкретної позиції. Таким чином за допомогою RelativeLayout ми отримуємо «гумовий», легко програмований інтерфейс. Конструктори ті ж самі, що і у LinearLayout.

TableLayout. Цей менеджер розміщення, розташовує елементи в табличному вигляді по

рядках і колонках. TableLayout складається з елементів розмітки TableRow, що визначають рядок в якому буде міститися віджет. TableLayout не відображує кордону осередків з яких складається кожен рядок. TableLayout має нуль або більше клітинок, а кожна клітинка може зберігати об'єкт для відображення (ViewObject). Число стовпців в таблиці визначається найдовшим за кількістю осередків рядком. Таблиця може містити порожні клітинки. Осередки можуть охоплювати декілька стовпців, також як і в HTML. Ширина колонки є максимальною за довжиною осередків, що належать цьому стовпцю.

У конкретних випадках, можна налаштувати таблицю під власні вимоги. У цьому допоможуть стандартні методи, котрі жорстко задають параметри таблиці.

TabLayout. Іноді нам доводиться описувати кілька різних відображень в одному вікні, забезпечуючи між ними швидке і комфортне перемикання. В такому випадку необхідно використовувати вкладку «менеджер розміщення» (tablayout), щоб створити tablayout. Необхідно виконати наступне: створити TabHost або у вигляді xml опису, або безпосередньо кодом в програмі. Цей елемент є контейнером для вкладок (табів) віконного виду. TabHost повинен мати два дочірніх елемента: TabWidget і FrameLayout.

TabWidget- елемент представляє список вкладок FrameLayout, який використовується для відображення вмісту кожної вкладки.

Існує два способи використання TabLayout:

- Можна створити кілька подань в одній Activity і перемикатися між ними, клацаючи по табам.
- Можна створити кілька Activity, що описують різні представлення, а в головному Activity забезпечити коректне перемикання.

Android SDK забезпечує великий набір інструментів, які необхідні для налаштування додатків. У вас повинен бути JDWP – сумісний відладчик (Java Debug Wire Protocol (JDWP) – це протокол, який використовується для обміну даними між відладчиком і віртуальною машиною Java, яку він і налагоджує.

Основні компоненти середовища налагодження Android:

ADB (AndroidDebugBridge) – виступає в якості посередника між пристроєм і програмою.

Dalvik Debug Monitor Service (DDMS) – це елемент плагіна Android, який використовується для аналізу роботи віртуальної машини. Послідовно відображає потік інформації, яка описує кожен крок роботи віртуальної машини.

Device or Android Virtual Device (AVD). Ваша програма повинна виконуватися на пристрої або в AVD так, щоб вона могла бути налагоджена. Демон adb пристрою працює на пристрої або емуляторі, і надає засоби для демона adb хоста для надання зв'язку з пристроєм чи емулятором.

ADB – клієнт – серверний додаток, що складається з трьох компонентів (рис. 1.2).

Клієнт запускається на машині розробника. Клієнт можна запустити з командного рядка за допомогою команд, що посилаються на ADB та інші інструменти Android, накшталт плагіна ADT і DDMS, теж створюють ADB клієнти.

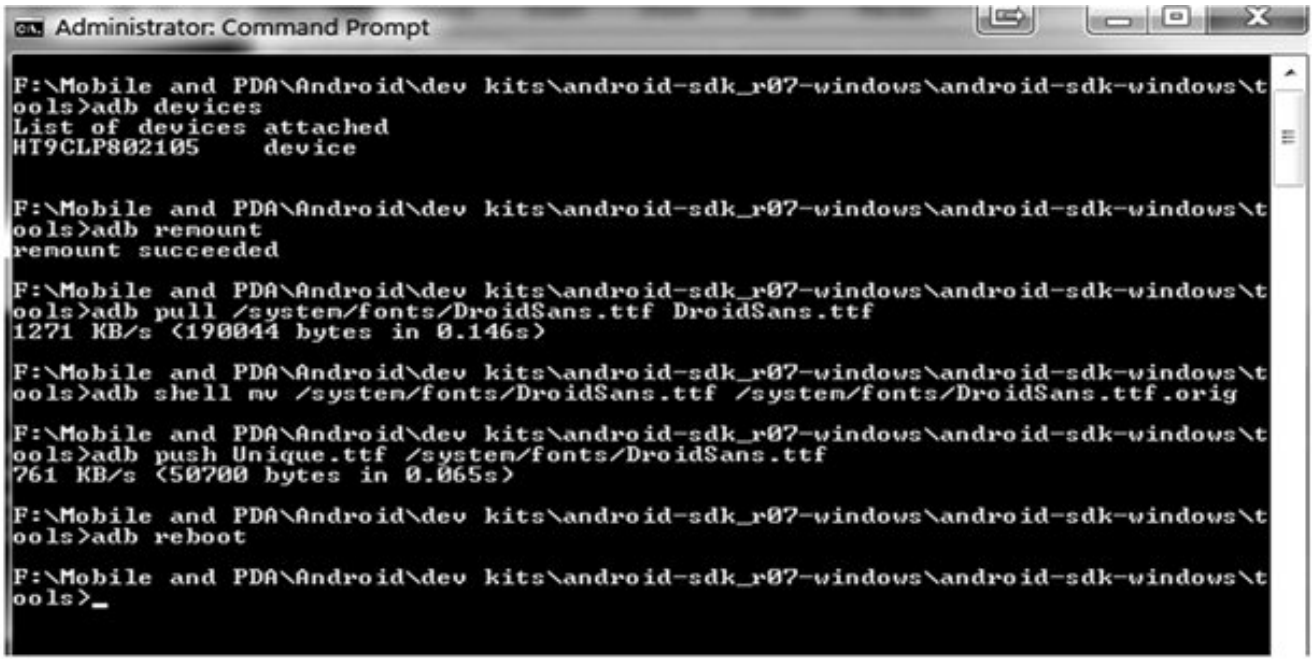

Рис. 1.2. Приклад використання клієнт – серверного додатка ADB

Сервер запускається на машині розробника у вигляді фонового процесу. Сервер управляє ADB-сервісом, запущеним на емуляторі або пристрої.

Сервіс – фоновий процес, який запускається на кожному емуляторі або пристрої.

Debug	logcat [<option>] [<filter – specs>]	Prints log data to the screen	
	bigreport	Prints dumpsys, dumpstate and logcat data to rhe screen, for the purposes of bug reporting.	
	jdwp	Prints a list of available JDWP processes on a given device.	You can use the forward jdwp:<Pid>port-forwarding specification to connect to a specific JDWP process. For example: adb forward tcp: 800 jdwp: 472 jdb -attach localhost: 8000

LogCat – система ведення логу в Android забезпечує механізм для збору і перегляду системних налагоджувальних повідомлень. Список з різних додатків і елементів системи Android збирається воєдино, а потім їх можна переглядати і фільтрувати за допомогою команди logcat (рис. 1.3).

Logcat можна використовувати для перегляду та стеження за вмістом буферів системного логу. Формат використання:

[Adb] logcat [<option>]... [<filter – spec>]...

Фільтрація виведення логу: кожне повідомлення логу в Android має теги пріоритет.

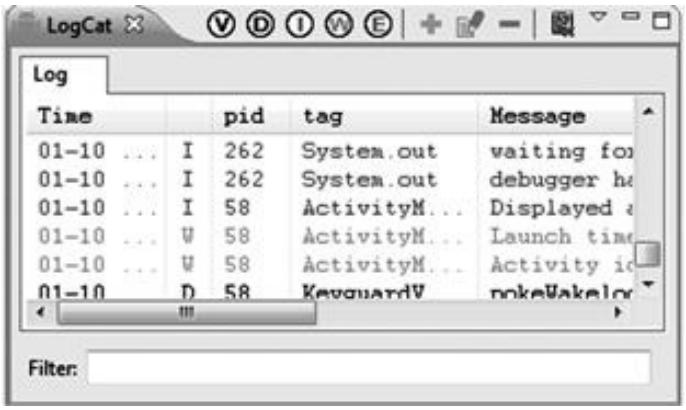

Рис. 1.3. Приклад використання LogCat

Тег – це рядок, який вказує на компонент системи, від якого прийнято повідомлення (наприклад, View для системи view).

Пріоритет має одне з нижченаведених значень (в порядку від меншого до більшого): V – Verbose (нижчий пріоритет); D – Debug; I – Info; W – Warning; E – Error; F – Fatal; S – Silent (найвищий пріоритет, при якому нічого не виводиться).

Вирази фільтра дозволяють вказати системі потрібні комбінації <тег> і <пріоритет>. Решта повідомлення система не виводить. Вирази фільтра мають наступний формат: <тег>: <пріоритет>.

Налагоджувати додатки в Eclipse набагато простіше, ніж в консолі, завдяки використанню standpoints. Але проблема налагодження через Eclipse може виникнути, якщо ви налагоджуєте додатки, запущені на емуляторі, а не на пристрої. Емулятор працює досить повільно, а при налагодженні кожен перехід працює більш 3 секунд. Це дуже довго. Так що, якщо ви налагоджуєте великі додатки і при налагодженні наткнулися на цикл, що містить 100 ітерацій, то, щоб його пройти, вам буде потрібно більше length (circle) * 3 * 100 секунд. У таких випадках краще використовувати налагодження з фільтрами через консоль і виводити інформацію про відладку у файл з подальшим виявленням помилок.

При налагодженні в Eclipse користуйтеся утилітою

LogCat (Window – > Show view – > Android – > LogCat),

яка, по суті, є оболонкою команд налагодження через консоль.

ОПИС ПРОГРАМИ

Додаток, який буде детально розглянуто нижче, написано в IDE Eclipse Indigo, тому використовуйте саме цю версію або більш пізню.

Приклад має виключно навчальний характер і спрямований на те, щоб показати розробнику процес ініціалізації опису внутрішнього устрою компонентів UI і дизайну інтерактивних додатків.

Загальна ідея. Зазвичай, якщо програма не має складної структури, то в одному стані додатку використовується тільки один стиль розміщення. Ми розглянемо додаток, який містить в собі багато різних layouts. Це досягається шляхом створення в якості головного layout, TabLayout, у якого кожен tab має свій спосіб розміщення. Звичай-

но, така архітектура використовується рідко, тому що користувачеві приємніше бачити плавне перемикання між табами на противагу нашому варіанту.

Структура.

TabLayout →
- Tab1: -> LinearLayout.
- Tab2: -> RelativeLayout.
- Tab3: -> GridLayout.

Архітектура Tab1.

Інтерфейс для роботи з датою в Android.

Компоненти :
- TextView – просте текстове поле, в якому буде відображатися поточна або вручну встановлена дата.
- Button – кнопка, яка ініціює появу віконця DataPicker, в якому відображається поточна дата.

У вікні стандартно є контролі, що дозволяють змінити дату.

Архітектура Tab2.

У цьому Табі будуть представлені деякі віджети UI і, як бонус, буде продемонстрований Spinner і процес взаємодії з ним.

Компоненти:
- TextView – текстове поле, яке відображає статичну інформацію.
- EditText – текстове поле з підтримкою редагування.
- Button – кнопка OK.
- Button – кнопка Cancel.
- Spinner – випадаючий список з обробкою події вибору.

Архітектура Tab3.

У цьому Табі будуть представлені графічні об'єкти (картинки), розміщені у вигляді таблиці.

Компоненти:
- GridView – представлення, що відображає елементи в двовимірній сітці з підтримкою вертикального або горизонтального скролу. Елементи, асоційовані з Grid-поданням, повинні бути представлені за допомогою ListAdapter.

Модель « Дизайн – XML»

Two hands. Android SDK дуже зручний не тільки тим, що представляє зрозумілий і прозорий API, який дозволяє навіть некваліфікованому розробнику швидко освоїти основні бібліотеки Android і приступити до написання власних програм, а й тим, що структура програми може бути розділена на дві незалежні частини: Java – код, що описує поведінку програми, і Xml – код, що описує компоненти UI (кнопки, списки, вікна).

Xml – підхід. Такий спосіб був обраний з кількох причин:
- Поділ роботи веб-дизайнера і програміста.

- Зменшення масивності програми за рахунок використання фрагментів xml – файлу в якості віджетів UI.

1.3 ВИКОНАННЯ РОБОТИ

Виконайте наступні кроки, щоб подивитися попередньо підготовлений додаток в дії.

Додаток може бути запущено з використанням віртуальної машини Davlik.

Імпорт програми.

Запустіть Eclipse, виберіть пункт меню «File», в списку виконайте команду «Import». У вікні у вкладці General, клацніть на «Existing project into workspace», в якому ви можете вказати шлях до проекту програми у вигляді архіву, або у вигляді простої папки. Приклад знаходиться в каталозі lab01 файлу labAtom32.rar. Потім натискаєте на кнопку «OK». в «Packageexplorer» у вас з'явиться проект з початковим кодом.

Запуск програми.

Натисніть на стрілку поруч із кнопкою «Run», у вспливаючому списку:

Run as – > Android application

В результаті дій проект перебудується і запуститься додаток. Після завантаження емулятора можна переконатися в працездатності програми.

Файл графічного інтерфейсу. Розглянемо докладніше файл опису графічного інтерфейсу додатку. Дуже часто при написанні додатків виникає необхідність розмістити на екрані кілька подань, абсолютно різних і непов'язаних один з одним логічно. В цьому випадку програміст відразу ж уявляє собі інтерфейс розташування вкладок браузера. Такий інтерфейс підтримує і операційна система Android.

В папці layouts проекту відкрийте за допомогою xml редактора файл «tab_layout.xml». Розглянемо його вміст.

Найперший вузол, «Tabhost», являє собою контейнер для вкладок. Він містить всього два типи елементів: «TabWidget» і «FrameLayout». Елемент «TabWidget» використовується для опису дизайну панелі вкладок, містить список вкладок. Менеджер розміщення «FrameLayout» необхідний при опису вмісту однієї вкладки. Для «TabHost» і «FrameLayout» задані ідентифікатори, так як в коді програми нам буде необхідно до них звертатися. Менеджер розміщення «LinearLayout» буде використовуватися на першій вкладці. Він містить текстове поле для відображення дати і кнопку, що дозволяє встановити дату. Наступний менеджер розміщення – «RelativeLayout»- містить кілька текстових полів: поле з підтримкою редагування, кілька кнопок, що випадають. І, нарешті, останній менеджер розміщення – «LinearLayout» – містить «Grid» представлення, для відображення картинок в таблиці. Кількість колонок жорстко не задано, але можна побачити, що ширина кожної колонки – 90 пікселів.

Код додатка. У файлі «LabAndroidActivity.java» знаходиться код, що описує всю логіку програми. Архітектура програми заснована на наступній ідеї: всередині однієї activity забезпечити навігацію між represent – tab використовуючи «TabHost».

Перш, ніж переходити до детального розбору коду, зупинимося на питанні «Життєвий

цикл додатку».

На відміну від додатків в більшості інших систем додатки Android не мають єдиної точки входу (наприклад, відсутня функція main ()). Замість цього існують чотири типи основних компонентів, з яких будуються андроїд – додатки і які система може запускати в міру необхідності. Це Activity, Service, Broadcast receiver і Content provider. Як згадувалося раніше, Activity є візуальним інтерфейсом для однієї дії, яку користувач може зробити. Додаток зазвичай будується з декількох Activity, кожен з яких виконує свою певну функцію. Зазвичай Activity не залежать одне від одного. Activity може перебувати в одному з трьох станів:

- Активно (active) або запущено (running) – знаходиться на передньому плані (на вершині стека activity поточного завдання) і має фокус для взаємодії з користувачем.
- Припинено (paused) – втратило фокус, але все ще видно користувачеві. Зверху знаходиться інше Activity, яке або прозоро або закриває не весь екран. Припинене activity повністю «живе» (його стан збережено і воно прив'язане до віконного менеджера), але може бути знищено системою в разі нестачі пам'яті.
- Зупинено (stopped) – повністю перекрито іншим activity. Воно більше не видно користувачеві і буде знищено системою, коли знадобиться пам'ять.

Життєвий цикл Activity складається з трьох вкладених циклів:

- Життєвий цикл Activity починається з виклику методу onCreate (), в якому проводиться первісна настройка глобального стану, і завершується викликом методу onDestroy (), в якому воно звільняє зайняті ресурси. Наприклад, в onCreate () можна створити потік, що завантажує дані з мережі у фоновому режимі, і потім зупинити його в onDestroy ().
- Видима частина життєвого циклу відбувається між викликами onStart () і onStop (). Протягом цього часу користувач може бачити Activity на екрані, хоча воно може бути не на передньому плані і не взаємодіятиме з користувачем. Між цими двома методами ви можете виділяти ресурси, необхідні для відображення Activity користувачеві. Методи onStart () і onStop () можуть викликатися стільки раз, скільки activity стає видимим або прихованим для користувача.
- На передньому плані Activity знаходиться між викликами onResume () і onPause (). Протягом цього часу Activity знаходиться поверх інших і взаємодіє з користувачем. Activity може часто переходити в стан паузи і виходити з неї. Наприклад, метод onPause () може бути викликаний, коли пристрій переходить в сплячий режим або коли запускається інше Activity, а метод onResume () – при отриманні результату закривається Activity. Таким чином, код в цих двох методах повинен бути досить легким.

Розглянувши основні деталі побудови програми, перейдемо до огляду коду. У головному методі onCreate (), задамо в якості вмісту Activity створений нами інтерфейс за допомогою виклику методу setContentView («Шлях до файлу через клас ресурсів»). Дуже часто при роздільному описі інтерфейсу і логіки додатка, необхідно отримати доступ до конкретного елементу інтерфейсу (кнопки), щоб призначити йому певну функцію. В

такому випадку, найкращим способом є використання методу findViewById («Ідентифікатор»). Параметр «Ідентифікатор» – це ім'я компонента визначене у файлі інтерфейсу. Використання методу setup (), необхідно перед додаванням вкладок, якщо об'єкт «TabHost» завантажується через файл, як в нашому прикладі.

Далі використовується TabSpec для створення вкладки, встановлюється індикатор і вміст вкладки. У класі визначена закрита змінна mDateDisplay, в якій зберігається час, і змінна mPickDate- кнопка, що викликає діалог DatePicker. У кнопки встановлено фон з авто – тригером, що підключається з файлу «Button.xml». Кнопці призначений слухач на натискання, всередині визначення якого реалізований метод onClick (View), що викликається щоразу при виникненні події. У методі onClick () викликається функція ShowDialog (), яка тягне функцію onCreateDialog (). Зверніть увагу, що всередині цієї функції ініціалізується об'єкт DatePickerDialog, причому другим параметром конструктору передається об'єкт «зміна дати». В змінні mYear, mMonth, mDay заноситься поточна дата і викликається метод UpdateDisplay (), який оновлює вміст текстового блоку.

Аналогічно створюється друга вкладка і заповнюється вмістом. Єдине, що варто тут відзначити, це клас Spinner. Об'єкт класу Spinner – випадає вниз списку, з можливістю вибору елементів. У файлі strings.xml визначено тег string – array, що містить елементи item. По суті, там містяться дані, які будуть відображені у Спиннері. Метод createFromResource (), створює новий ArrayAdapter, який пов'язує кожен елемент масиву «planets_array» з початковим поданням Спиннера. Інакше кажучи, як кожен елемент буде відображатися в той момент, коли користувач надумає розкрити Спиннером. Метод setDropDownViewResource встановлює налаштування «розкриття – вниз» для Спиннера, і, нарешті, всі елементи ArrayAdapter зіставляються з Спиннером за допомогою методу setAdapter (). Для Спиннеру реєструється слухач на подію «Вибір елемента списку». Реалізація об'єкту клас MyOnItemSelectedListener реалізує інтерфейс OnItemSelectedListener. У методі onItemSelected () користувачеві показується спливаюче віконце з інформацією про вибраний елемент.

Остання вкладка відображає елементи за допомогою табличного менеджера компоновки, описаного в файлі інтерфейсу. За допомогою методу setAdapter () призначаються дані для відображення. ImageAdapter- дочірній клас BaseAdapter. Масив mThumbs містить посилання на картинки через клас ресурсів. Аналогічно як і Спіннеру, GridView встановлюється слухач на подію «Вибір елемента». Закінчують метод onCreate () три рядки коду, в яких контейнеру табів додаються створені по черзі вкладки.

1.4 ВИСНОВКИ

Матеріал цієї лабораторної роботи № 1 допоможе вибрати, завантажити і проінсталювати необхідне програмне забезпечення і почати з ним працювати для програмування мобільного пристрію.

ЛАБОРАТОРНА РОБОТА № 2. РОЗРОБКА ПРОГРАМИ

Приклад отримання і запису даних від відеопристрою.

2.1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Презентацію до лабораторної роботи Ви можете завантажити тут.

Демонстрація процесу розробки практичного застосування, що дозволяє отримувати фотографії з вбудованої фотокамери і записувати їх в задану нами папку на зовнішню пам'ять мобільного пристрою.

Розглянемо найпростішу задачу – відеоспостереження, яке полягає в отриманні і передачі серії зображень з камери, встановленої на БПЛА, в ЦОД через GSM/GPRS канал.

2.2 ІНСТРУКЦІЯ ПО ВИКОНАННЮ ЛАБОРАТОРНОЇ РОБОТИ

Завдання відеоспостереження природним чином розпадається на складові частини (рис. 2.1). По-перше, нам потрібно вміти підключатися до Інтернету за допомогою GSM-модему. По-друге, ми повинні навчитися отримувати кадр від камери і зберігати його в форматі JPG. По-третє, нам потрібен механізм для прийому даних на ЦОД. Ці підзадачі незалежні один від одного.

Одним із стандартних датчиків отримання даних із зовнішнього середовища на різних мобільних пристроях – фотокамера. Зараз такі камери мають мініатюрний розмір, високу роздільну здатність фото і відео зйомки. В основному такі камери були доступні тільки як вбудована опція в мобільних пристроях. Останнім часом все частіше зустрічаються цифрові мініатюрні фотокамери, як самостійний пристрій або зовнішнє доповнення. ОС Android володіє всіма необхідними бібліотеками для роботи з фотокамерами, а також, для роботи з пам'яттю, наприклад, microSD, або частина вбудованої флеш – пам'ятті пристрою.

У цій лабораторній роботі ми будемо використовувати вбудовану фотокамеру і внутрішню пам'ять, подивимося, як підключати необхідні нам пристрої в додаток і яким чином використовувати потрібні нам бібліотеки в коді. Для цього будуть потрібні необхідні базові знання мови Java, знайомство с Android SDK і необхідними методами і класами.

В якості однієї зі складових програми будуть використані матеріали лабораторної роботи «Захоплення відеозображенням» з першої частини курсу. Також необхідно мобільний пристрій (планшет, комунікатор, нетбук і т. ін.), що працює під управлінням ОС Android. Нами використовувався планшет компанії Intel з ОС Android 4.0.

Завдання складається з трьох частин – отримання кадру від камери, стиснення кадру в формат jpeg, запис отриманого кадру в пам'ять мобільного пристрою.

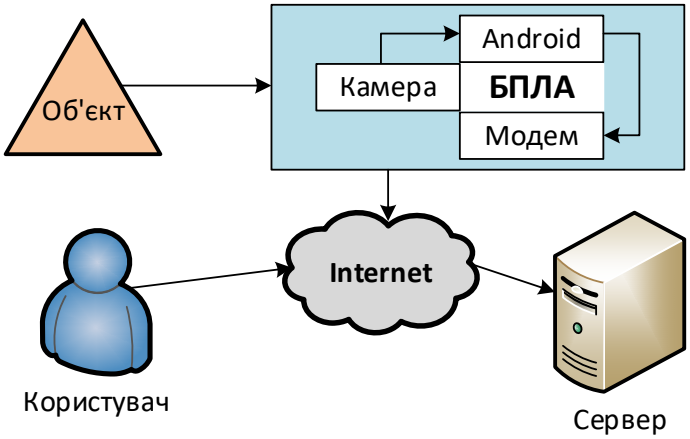

Рис. 2.1. Складові частини системи відеоспостереження

Для початку роботи необхідно запустити Eclipse SDK, створити новий робочий простір (наприклад, android_camera), і створити новий проект Android Application Project. Для цього необхідно зайти у вкладку File і перейти в New – > Android Application Project. Далі необхідно задати назву програми, версію Android і запустити проект. Ми будемо працювати з уже створеним нами проектом ImageUploader. Для цього необхідно увійти File – > Import – > і вибрати папку з проектом.

Далі опишемо, які необхідні елементи програмного коду нашого прикладу відносяться безпосередньо до роботи з фотокамерою і пам'яттю.

Для початку в проекті необхідно прописати те обладнання, яке буде задіяно в додатку і його додаткові функції. Для цього необхідно відкрити файл AndroidMonifest.xml і внести наступні рядки:

1. <uses – sdk

2. android: minSdkVersion = «8»

3. android: targetSdkVersion = «14» />

4. <uses – permission android: name = «android.permission.CAMERA» />

5. <uses – permission android: name = «android.permission.WRITE_EXTERNAL_STORAGE» />

6. <uses – feature android: name = «android.hardware.camera» />

7. <uses – feature android: name = «android.hardware.camera.autofocus» />

У першому, другому і третьому рядках описуються версії мінімальної SDK і тієї, для якої призначено додаток. Далі необхідно встановити «дозволи» на використання того чи іншого пристрою – permission. Рядки 4 та 5 є запитами на «дозволи» для користування без зайвих витрат і зовнішнього носія. У рядках 6 і 7 прописані функції камери. Якщо ви плануєте, що додаток буде працювати з відповідним обладнанням та його певними функціями, то функції вказуються без додаткових умов. Якщо ж ви хочете щоб ваш додаток також запускався на пристроях, які не мають зазначених пристроїв і функцій, то у відповідному рядку необхідно прописати:

android: required = «false» />

після вказівки функції. В даному прикладі ми хочемо скористатися камерою і функцією автофокус.

Для того щоб почати використовувати камеру, необхідно отримати доступ до неї. Для цього необхідно скористатися Camera.open(). У нашому прикладі частина коду роботи з камерою така:

```
1. public boolean openCam () {
2. try {
3. camera = Camera.open ();
4. Camera.Parameters parameters = camera.getParameters ();
5. parameters.setFlashMode (Camera.Parameters.FLASH_MODE_OFF);
6. if (focusBox.isChecked ())
7. parameters.setFocusMode (Camera.Parameters.FOCUS_MODE_AUTO);
8. else
9. parameters.setFocusMode (Camera.Parameters.FOCUS_MODE_INFINITY);
10. parameters.setJpegQuality (95);
11. List <Camera.Size> psizes = parameters.getSupportedPictureSizes ();
12. int w = 0;
13. int h = 0;
14. for (int i = 0; i <psizes.size (); i ++) {
15. if (w <psizes.get (i).width && h <psizes.get (i).height) {
16. w = psizes.get (i).width;
17. h = psizes.get (i).height;
18.}
19.}
20. log ( «Picture size:» + String.format ( «% dx% d», w, h));
21. parameters.setPictureSize (w, h);
22. camera.setParameters (parameters);
23. camera.setPreviewDisplay (fakeView.getHolder ());
24.} catch (Exception e) {
25. e.printStackTrace ();
26. log ( «Camera open fail:» + e.toString ());
27. closeCam ();
28. return false;
29.}
30. return true;
31.}
```

На початку ми вказуємо на ініціалізацію і початок роботи з камерою. У рядках з 4 по

23 ми вказуємо параметри роботи камери і параметри одержуваних нами даних в залежності від того, які дані нам необхідні. У нашому прикладі це: робота без спалаху, з автофокусом, дані в форматі jpeg. Потім ми перебираємо розміри зображення, які може надати камера, і вибираємо максимальний. В кінці ми встановлюємо камері фіктивний Surface для попереднього відображення кадру. Для вирішення завдання він не потрібен, але камера не зможе без нього працювати. При неправильній роботі камери або її відсутності, програма повідомляє про це «Camera open fail» і виводить подробиці помилки.

Далі в прикладі описується ще дві функції камери – зупинка роботи камери closeCam() і takePhoto(). Ці класи викликаються засобом натискання заданих раніше кнопок. Сам код такий:

```
1. public void closeCam () {
2. if (camera! = Null) {
3. camera.stopPreview ();
4. camera.release ();
5. camera = null;
6. log ( «Release camera»);
7. }
8. }
9.
10. private void takePhoto () {
11. if (previewStarted) {
12. log ( «Skip photo, wait previous...»);
13. return;
14.}
15. if (camera == null) {
16. log ( «Camera not initialized...»);
17. return;
18.}
19. previewStarted = true;
20. camera.startPreview ();
21. camera.takePicture (null, null, jpegCallback);
22.}
```

де з 1 по 8 рядок – команда зупинки роботи камери, а з 9 по 22 – команда зробити фото в форматі jpeg. Зверніть увагу, що для отримання кадру від камери безпосередньо перед цим включається режим попереднього перегляду. Потім, в обробнику кадру, значок попереднього перегляду буде скинутий.

Далі необхідно вказати, що робити з даними отриманими з камери. У нашому прикладі, ми конвертуємо дані в формат jpeg з різними параметрами «small» і «big»», даємо їм імена і записуємо їх в дві різні папки. Ми робимо два розміри для отримання фотографій попереднього перегляду, а також для можливої їх відправки через Internet.

1. PictureCallback jpegCallback = new PictureCallback () {

2. public void onPictureTaken (byte [] data, Camera camera) {

3. FileOutputStream outStream = null;

4. String filename = String.format (imageDir + File.separator + «big» + File.separator + «% d.jpg»,

5. System.currentTimeMillis () / 1000);

6. String filename2 = String.format (imageDir + File.separator + «small» + File.separator + «% d.jpg»,

7. System.currentTimeMillis () / 1000);

8. String gps = «No»;

9. try {

10. outStream = newFileOutputStream (filename);

11. outStream.write (data);

12. outStream.close ();

13. Bitmap origImage = BitmapFactory.decodeByteArray (data, 0, data.length);

14. int origHeight = origImage.getHeight ();

15. int origWidth = origImage.getWidth ();

16. double scale = (origHeight> origWidth? OrigHeight: origWidth) / 300;

17. int newHeight = (int) (origHeight / scale);

18. int newWidth = (int) (origWidth / scale);

19. Bitmap smallImage = Bitmap.createScaledBitmap (origImage, newWidth, newHeight, false);

20. origImage.recycle ();

21. int quality = 75;

22. outStream = new FileOutputStream (filename2);

23. BufferedOutputStream bos = new BufferedOutputStream (outStream);

24. smallImage.compress (CompressFormat.JPEG, quality, bos);

25. smallImage.recycle ();

26. bos.close ();

27.};

У рядках з 1 по 8 описується місце, де необхідно брати вихідні дані зображення і яким чином формувати назву файлу. У рядках з 9 по 27 вказується, що необхідно зробити

з даними і куди їх записати, а саме створюється копія зображення меншого розміру зі зниженою якістю для більш швидкої передачі по мережі. Наприклад, вихідні дані беруться з камери, назва формується виходячи з номера фотографії, і записується на карту пам'яті у форматі .jpg.

2.3 ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Змінити папку для збереження і розмір одержуваного кадру.

Додати використання спалаху під час зйомки.

Додати функцію визначення особи на кадрі.

2.4 ВИСНОВКИ

У лабораторній роботі розглянуто приклад процесу розробки практичного застосування під ОС Android, розглянутий приклад розробки прототипу практичного застосування по захопленню відеокадрів, збереженню їх і записи в форматі.jpg на зовнішній носій.

ЛАБОРАТОРНА РОБОТА 3. ОЗНАЙОМЛЕННЯ З JADE

Знайомство з мультиагентною платформою JADE, установка платформи на робочий комп'ютер, освоєння протоколу передачі даних між агентами на платформі, запуск головного контейнера платформи і нового агента на прикладі агентів «Ping Agent» – запит – відповідь.

3.1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Презентацію до лабораторної роботи Ви можете завантажити тут.

Познайомитися з мультиагентною платформою JADE, встановити платформу на робочий комп'ютер, освоїти протокол передачі даних між агентами на платформі, запустити головний контейнер платформи і нового агента на прикладі агентів «Ping Agent» – запит – відповідь.

3.2 ІНСТРУКЦІЯ ПО ВИКОНАННЮ ЛАБОРАТОРНОЇ РОБОТИ ПІДГОТОВКА

Перед початком роботи необхідно запустити середовище розробки Eclipse IDE, завантажити і встановити JDK і JADE. Запустити платформу можна двома способами: з командного рядка або в середовищі розробки. Як і в попередніх лабораторних роботах, ми будемо використовувати середовище розробки Eclipse IDE. Для запуску з командного рядка необхідно звернутися до інструкції на офіційному сайті платформи

JADE за посиланням :http://jade.tilab.com/doc/index.html.

Для початку знайомства з JADE необхідно встановити JDK і запустити Eclipse IDE (див. лаб. роботу 1). В Eclipse IDE створити робочу папку і відкрити новий Java проект. Далі відкрити новий файл Java, в який будемо записувати програмний код агента.

Перед тим як створювати агента, нам буде потрібно встановити платформу JADE. Завантажити архів з платформою можна з офіційного сайту JADE за посиланням: http://jade.tilab.com/. Необхідно буде зареєструватися і отримати підтвердження реєстрації електронною поштою. Скачайте останню версію платформи. Розпакуйте архів і відкрийте його.

Архів містить 3 папки:
- JADE – bin – 4.X – папка з виконуваними файлами;
- JADE – bin – 4.X – папка з виконуваними файлами;
- JADE – example – 4.X – папка з прикладами.

Створіть окрему папку, наприклад, з назвою Jade, і скопіюйте вміст всіх трьох папок з архіву в нову папку (для спрощення адреси, найкраще нову папку створити в корені диска).

В результаті ми отримуємо готову платформу. Перевірьте розташування таких файлів як:

- build.xml – знаходиться в корені папки Jade (створеної нами);
- jade.jar – знаходиться в папці lib/;
- папки вихідних файлів examples/і jade/ – знаходяться в src/.

Перед запуском платформи і подальшої роботи з нею хотілося б визначити такі моменти в архітектурі системи. Основними елементами мультиагентної системи є Агенти. Мультиагентна платформа складається з контейнерів (модулів), в яких «живуть» Агенти. Контейнер (Модуль) може не містити жодного агента або містити скільки завгодно агентів. Кожен контейнер (модуль) пов'язаний мережевим адресою та іменем. Є особливий контейнер (модуль) – головний контейнер (модуль) – без нього платформа не працює. Головний контейнер (модуль) має кілька особливостей. Перерахуємо їх:

- він повинен бути створений першим;
- він повинен включати в себе двох спеціальних агентів:

AMS (agent management system) – агент, який управляє іншими агентами;

він здатний створювати і зупиняти агентів;

DF (directory facilitation) – по суті представляє з себе жовті сторінки, якому записуються адреси, імена та

можливості агентів в системі.

Виконання роботи:

Для запуску платформи необхідно запустити Eclipse IDE, створити нове робоче середовище і в ньому відкрити новий Java проект (рис. 3.1).

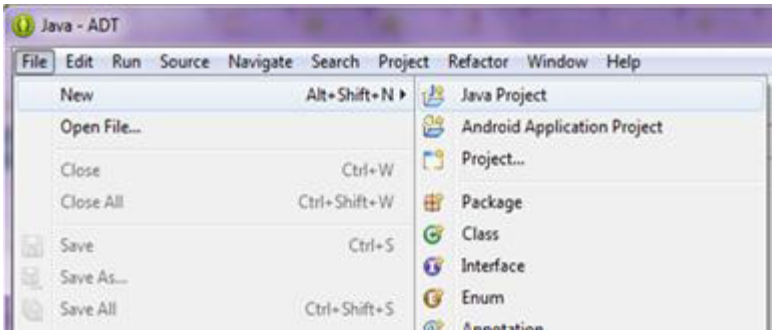

Рис. 3.1. Новий Java проект в Eclipse IDE

Далі переходимо у вікно створення проекту , задаємо Next. У наступному вікні необхідно підключити до нашого проекту бібліотеки Jade (рис. 3.2).

Для підключення бібліотек необхідно натиснути Add External JARs і вказати шлях до бібліотек. Вони знаходяться в папці \ lib, куди були скопійовані всі файли архівів платформи Jade. Після цього, натискаємо Finish.

Тепер до проекту підключені бібліотеки JADE. Створюємо нового Агента з прикладу «PingAgent». Для цього необхідно створити новий клас в папці \ src (рис. 3.3).

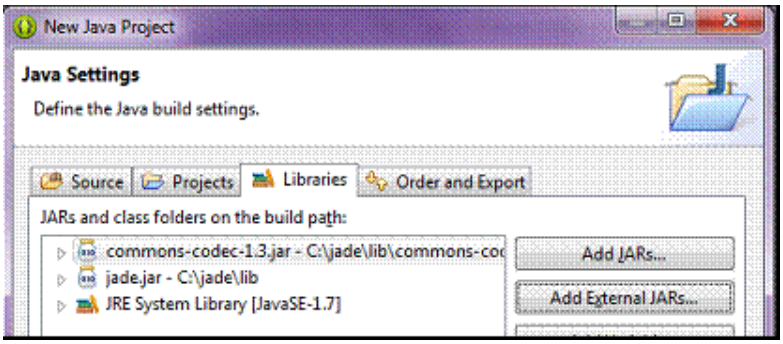

Рис. 3.2. Бібліотеки Jade в проекті

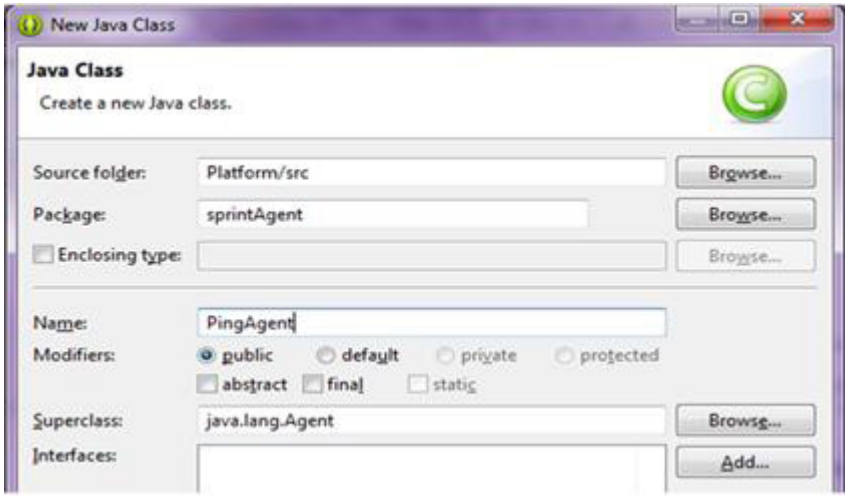

Рис. 3.3. Новий клас

Тепер вводимо інформацію, створюємо новий клас в папку Project / src в пакеті sprintintel з іменем PingAgent. Також на цьому етапі необхідно записати в поле «Superclass» наступну інформацію: java.lang.Agent і натиснути Finish.

Зараз у нас є новий Агент і підключені необхідні бібліотеки. Необхідно поставити програмний код Агента. Для цього відкриваємо PingAgent.java з папки examples \ PingAgent, яка знаходиться в директорії, в яку розпаковані архіви Jade. Копіюємо весь вміст прикладу в нашого Агента. Подивимося на код нашого Агента, в ньому є помилки, які необхідно виправити (рис. 3.4).

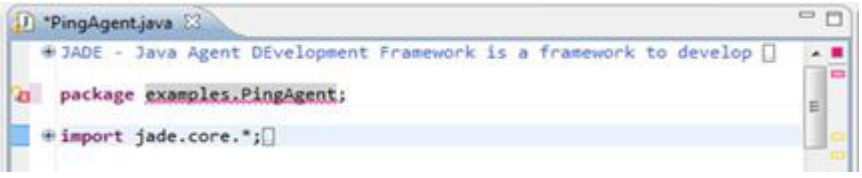

Рис. 3.4. Помилка в коді Агента

Ця помилка цілком зрозуміла. Необхідно виправити ім'я пакету, який ми використовуємо. У нашому випадку – це sprintintel. Далі необхідно обов'язково зберегти зміни у файлі Агента.

Для того щоб запустити Агента і мультиагентну платформу, переходимо у меню Run Configuration. У ньому додаємо нову конфігурацію запуску Java – додатку. На рис. 3.5 і рис. 3.6 показані параметри головного класу (MainClass) – jade.Boot і видно назву

справжнього проекту. В поле Program Argument вкладки Arguments вводимо значення « – gui». Таким чином, ми визначили, що нам необхідно запустити платформу і головний контейнер, який вже містить додатковий клас нашого агента PingAgent.

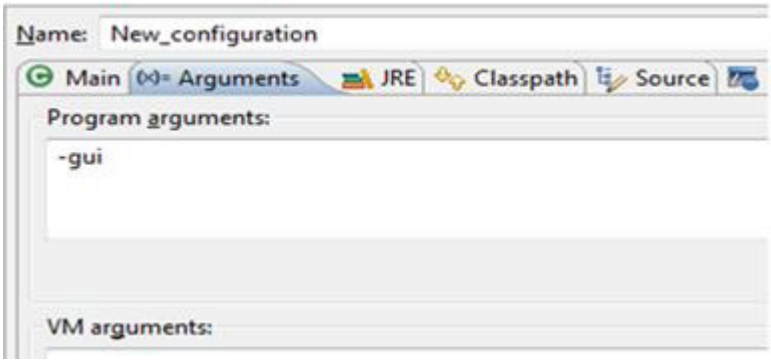

Рис. 3.5. Конфігурація запуску, вкладка Arguments

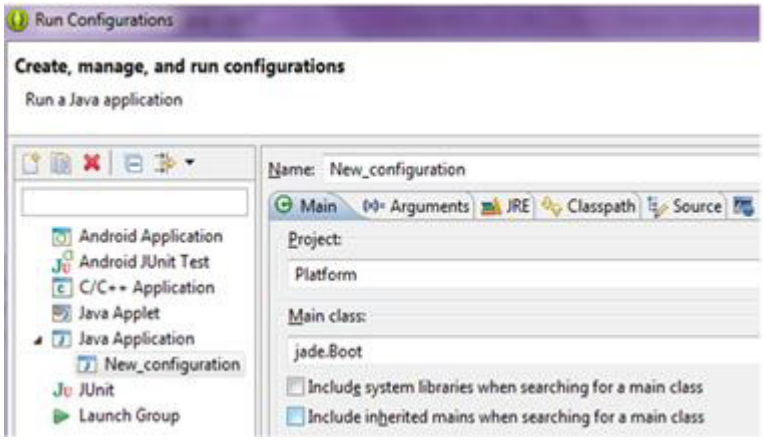

Рис. 3.6. Конфігурація запуску. вкладка Main

Натискаємо на кнопку Run і запускаємо нашу платформу. Платформа запущена, і ми бачимо призначений для користувача інтерфейс, в якому запущена платформа і два базових агента AMS і DF. Щоб додати в головний контейнер нашого Агента « Ping», необхідно увійти у вкладку Action і додати клас нашого Агента (рис. 3.7).

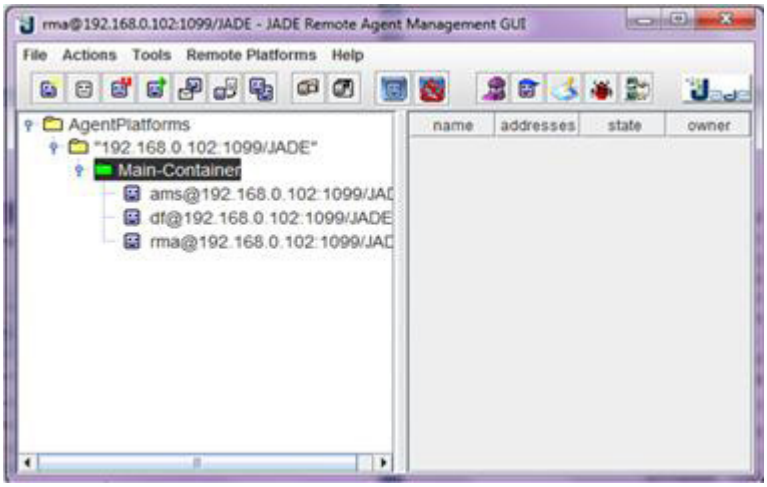

Рис. 3.7. Інтерфейс користувача на платформі Jade

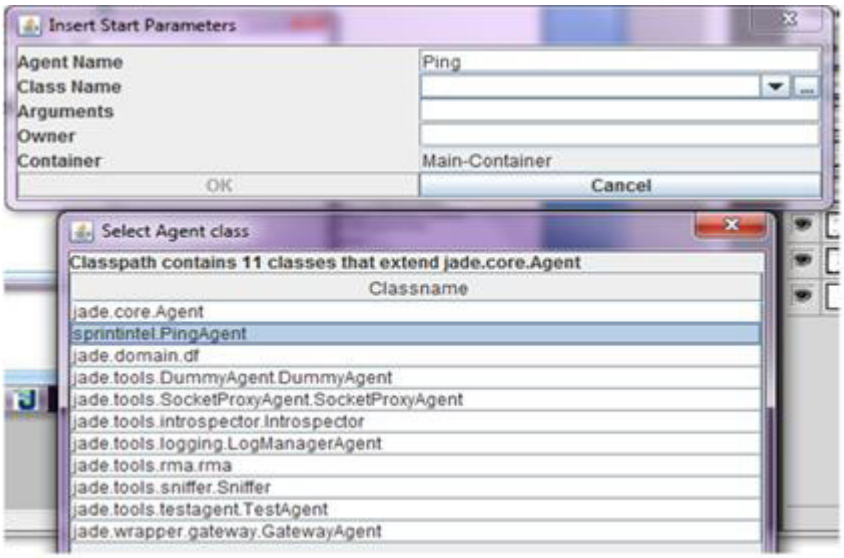

Рис. 3.8. Class Агента Ping Agent в пакеті sprintintel

Наш агент тепер знаходиться в головному контейнері. У чому ж суть Агента в прикладі PingAgent? Основне завдання цього агента – це відправляти відповідь у вигляді слова «Ping» при запиті у вигляді слова «Pong». У коді ця частина виглядає так:

```
 public void action () {
  ACLMessage msg = myAgent.receive ();
  if (msg! = null){
    ACLMessage reply = msg.createReply ();
    if (msg.getPerformative () == ACLMessage.REQUEST) {
String content = msg.getContent ();
if ((content! = null) &&
     (content.indexOf ( «ping»)! = – 1)) {
  myLogger.log (Logger.INFO, «Agent
 «+ GetLocalName () +» – Received PING Request
from «+ msg.getSender (). getLocalName ());
  reply.setPerformative (ACLMessage.INFORM);
  reply.setContent ( «pong»);
      }
    }
  }
}
```

При запуску агента в головному контейнері він звертається в «жовті сторінки» – агент DF, який записує у себе адресу цього агента. Всі запити здійснюються через DF агента. Таким чином, в систему можуть входити і виходити нові агенти і бути «на зв'язку».

Тепер перевіримо роботу нашого Агента і відправимо йому запит за допомогою

DummyAgent – готовий агент з призначеним для користувача інтерфейсом, за допомогою якого можливо відправляти повідомлення на інші Агенти і отримувати від них відповіді. Для запуску DummyAgent необхідно натиснути кнопку StartDummyAgent в ряду кнопок швидкого доступу інтерфейсу користувача платформи. У самому Агенті необхідно скласти запит в правильній формі, як це показано на рис. 3.9

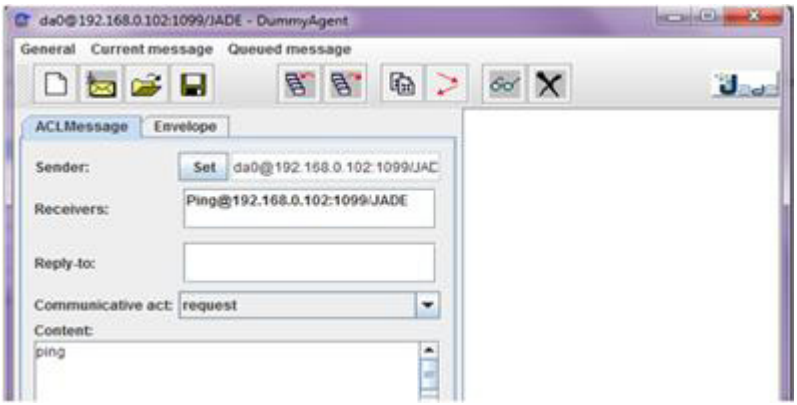

Рис. 3.9. DummyAgent, запит на Ping агента

Після правильного заповнення запиту необхідно натиснути на кнопку: відправити повідомлення. DummyAgent дозволяє, відправляє запит з одної точної адреси агента на точну адресу іншого агента навіть якщо вони знаходяться в різних контейнерах. Але ці агенти повинні бути прописані в одній мультиагентной системі (тобто відомі DF агентам). Справа, натиснувши на значок окуляри, в білому полі призначеного для користувача інтерфейсу DummyAgent, можна побачити, чи прийшла відповідь і якого вона змісту (рис. 3.10).

Для того, щоб простіше було відслідковувати відправку повідомлень, можна запустити ще одного агента Sniffer, у якому відображаються стрілками надіслані повідомлення між агентами (рис. 3.11).

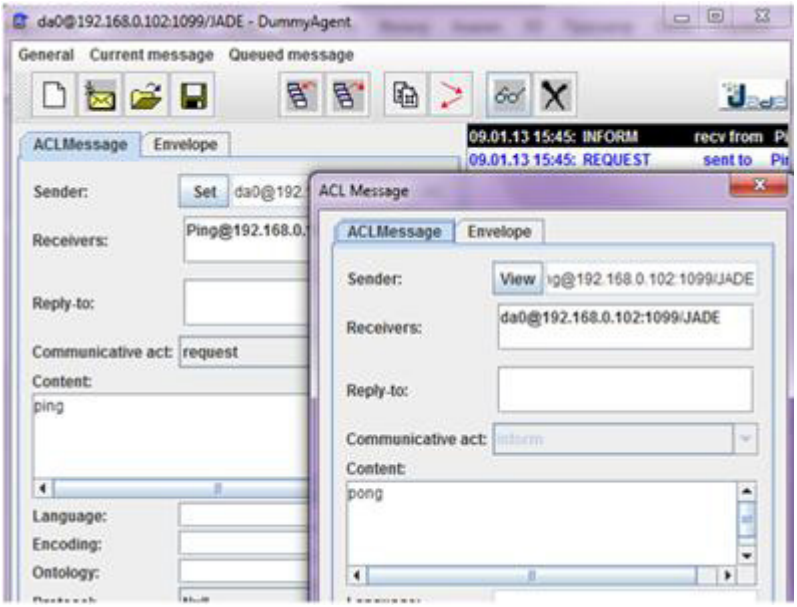

Рис. 3.10. Відповідь на запит DummyAgent

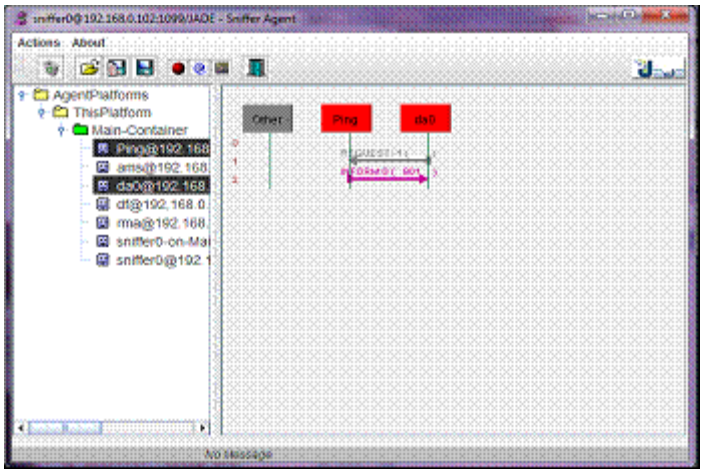

Рис. 3.11. Агент Sniffer

3.3 ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Створити ще одного агента Ping1 і відправити запит двом агентам одночасно.

Змінити текст повідомлення запиту і відповіді.

Створити Container1 (не головний) і створити в ньому агента.

Запустити агентів в прикладі папки src / examples / bookTrading.

3.4 ВИСНОВКИ

У лабораторній роботі були описані основні принципи розробки мультиагентних систем і розібраний приклад роботи з платформою JADE.

ЛАБОРАТОРНА РОБОТА 4. ПРИКЛАД МУЛЬТИАГЕНТНОГО ДОДАТКИ З JADE – АГЕНТОМ НА ПЛАТФОРМІ ANDROID

Створення декількох робочих контейнерів в мультиагентній платформі JADE, визначення взаємозв'язку між декількома контейнерами.

4.1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Презентацію до лабораторної роботи Ви можете завантажити тут.

Створити кілька робочих контейнерів в мультиагентній платформи JADE, визначити взаємозв'язок між декількома контейнерами, перенести один з контейнерів на мобільний пристрій, запустити додаток в Android emulator, запуск DummyAgent на мобільному пристрої з ОС Android.

4.2 ІНСТРУКЦІЯ ПО ВИКОНАННЮ ЛАБОРАТОРНОЇ РОБОТИ

ПІДГОТОВКА

Перед початком роботи необхідно запустити середовище розробки Eclipse IDE. Запустити платформу і агента PingAgent (див. лаб. раб. 3) Необхідно завантажити платформу JADE – LEAP Android (Lightweight Extensible Agent Platform) runtime environment. Також необхідно буде налаштувати Android віртуальний пристрій з підтримкою x86 архітектури для налагодження програми під наш планшет Intel.

Для того щоб отримати JADE – LEAP, необхідно перейти на офіційний сайт платформи JADE – http://jade.tilab.com/. Далі у вкладці « Add – ons» скачати повний архів Leap і JADE – Android у вигляді архівів. zip. Розпакуйте обидва архіва в папку , де знаходиться повний пакет файлів платформи JADE (див. лаб. раб. 3). У нашому прикладі ми отримуємо в папці \ jade додаткові папки\add – ons і\leap. У папці add – ons обов'язково повинні знаходитись підпапки:

- doc – містить документацію;
- lib – містить архів. jar, в якому знаходяться зібрані класи;
- src – ресурсний файл , якого немає в Jade Leap;
- demo – містить проект DummyAgent.

ВИКОНАННЯ РОБОТИ

Для запуску JADE-LEAP в Windows необхідно прописати параметр JAVA_HOME. Якщо він у вас ще не прописаний, зробити це можна перейшовши у вкладку Комп – > Властивості системи, далі перейти в Додаткові параметри системи в лівій частині з. Потрібна нам директорія Системні змінні знаходиться в опції Змінні середовища. Створюємо нову Системну змінну з назвою JAVA_HOME і прописуємо шлях в JRE. В нашому прикладі шлях до необхідних нам файлів наступний C:\Program Files \ Java \ jre7.

У нашому прикладі ми хочемо запустити агента на планшеті Intel з процесором Intel Atom. Для тестування програми зручно використовувати Android Virtual Device. Надалі, на цьому віртуальному пристрої можна буде запустити ще одного агента в системі.

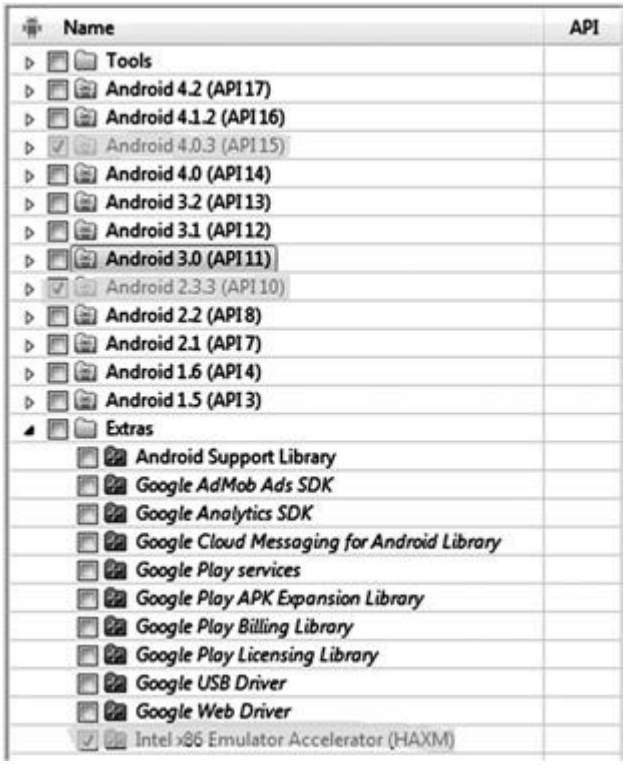

Рис. 4.1. Установка необхідних компонент

Для настройки віртуального пристрою необхідно перейти в Android SDK Manager і встановити всі елементи до версій Android 2.3.3 і 4.0.3, обов'язково провірити, що у версії 4.0.3 в ці елементи входить Intel x86 Atom System Image (рис. 4.1). Саме з версії 4.0.3 почалася офіційна підтримка Intel x86 Atom System Image. А також, нам знадобиться Intel x86 Emulator Accelerator (HAXM). The Intel Hardware Accelerated Execution Manager (Intel ® HAXM) – апаратна підтримка візуалізації, яка використовує технології Intel Virtualization Technology (Intel ® VT) для прискорення емуляції Android.

Далі потрібно перейти в Android Virtual Manager, де ми створимо новий віртуальний пристрій для тестування нашого додатку. Увійшовши в меню Android Virtual Manager, створюємо новий пристрій з необхідними нам параметрами (рис. 4.2). Так як ми будемо створювати додаток під планшет Intel, то бажано створити віртуальний пристрій з наближеними параметрами.

Пропишемо необхідні параметри віртуального пристрою на рис. 4.2. В якості імені можна ввести будь-яке значення. Наш пристрій ближче є планшетом з сенсорним екраном 10.1, вибираємо платформу Android 4.0.3. с API Level 15.

Галочки в графах Keyboard і Skin ми не встановлюємо. Camera Front і Back залишимо віртуальні, так як в нашому додатку камера поки не використовується, хоча в реальному пристрої вона є. Оперативну пам'ять встановлюємо 512Mb, а пам'ять, яка використовується програмою VM Heap-32. Такі параметри виявилися оптимальними для емуляції на нашому ноутбуці. Також встановлюємо внутрішню пам'ять 2000 Mb і

зовнішню 1000 Mb. Також ставимо галочку на Use Host GPU (Graphics processing unit) для прискорення роботи нашого емулятора. Залишилося тільки запустити Intelx86 Emulator Accelerator (HAXM) і можна працювати з емулятором під Intel Atom-ОС Android. Це можна зробити перейшовши в каталог C:\Program Files (x86)\ Android\ sdk\extras\intel\Hardware_Accelerated_Execution_Manager і запустивши IntelHaxm. exe. Тепер можна запустити наш віртуальний пристрій, перейшовши в меню Android Virtual Manager. Вибираємо наш пристрій і натискаємо start.

Контейнер платформи JADE запускається під час запису аргументу як – container. Щоб запустити контейнер в Eclipse IDE необхідно спочатку запустити Main Container. В нашому прикладі з головним контейнером компілюється і агент Ping (див. лаб. раб. 3). Після запуску Main Container треба перейти назад в проект Eclipse і запустити його з аргументом – container. Це можна зробити перейшовши у вкладку RunConfiguration (рис. 4.3).

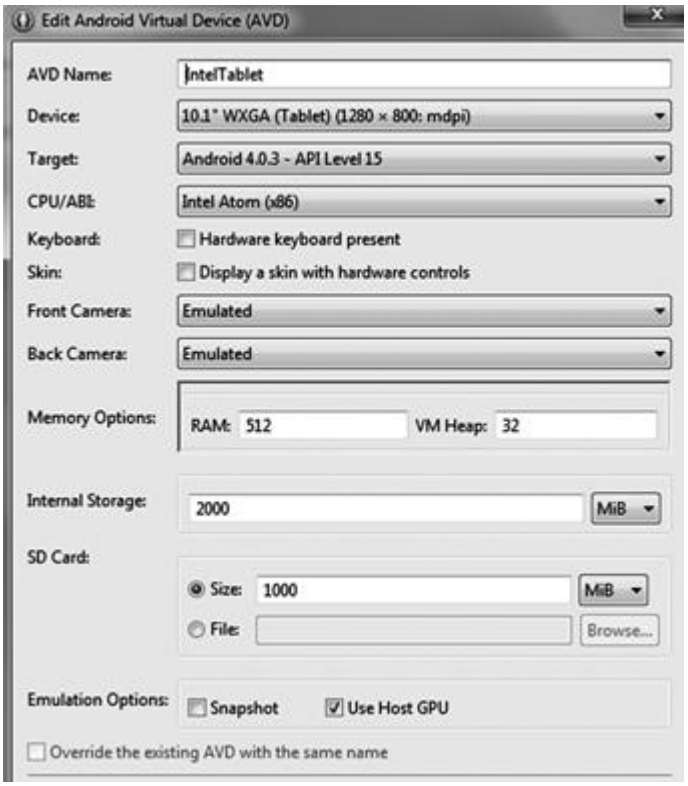

Рис. 4.2. Установки віртуального пристрою

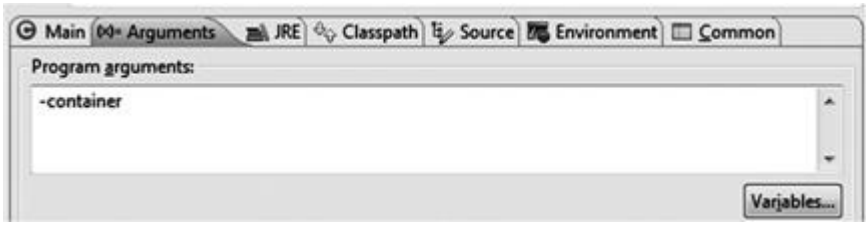

Рис. 4.3. Зміна Program argument

Після запуску в інтерфейсі платформи отримуємо додатковий контейнер Container – 1, який поки не містить агентів. Натиснувши лівою кнопкою миші на новий контейнер, перейшовши в StartNewAgent. запускаємо агента Ping (див. лаб. раб. 3).

Для того щоб передати просте повідомлення на створеного нами Ping агента, нам необхідно відкрити агента DummyAgent. Такий агент можна відкрити з мобільного пристрою з ОС Android. Перед тим як використовувати наш планшет, запустимо DummyAgent в налаштованому нами віртуальному пристрої. У відкритому робочому просторі необхідно імпортувати проект DummyAgent із скачаного нами раніше архіву Jade – Android. Заходимо у вкладку File – > Import – > General – > Existing Projects into Workspace. Знаходимо потрібний нам проект в папці jade/add-ons/jade4android/demo і натискаємо Import. Тепер необхідно додати потрібні бібліотеки Jade. Вони знаходяться в папці add – ons / jade4android /lib. Додати бібліотеки можна через вкладку Project – > Properties – > JavaBuild – > Library.

Для запуску агента на віртуальному пристрої перейдіть у вкладку RunConfiguration. Виберіть свій проект і віртуальний пристрій. Натисніть кнопку Apply, а потім Run. Запуститься віртуальний пристрій і DummyAgent. Тепер відправте повідомлення на створений нами Ping агент в Container – 1.

Після запуску програми ми можемо знайти готовий інсталяційний файл. apk в папці .bin нашого проекту.

4.3 ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Запустити інсталяційний файл на вашому пристрої.

Передати повідомлення на Ping агент.

Запустити готовий приклад ChatClient, скачавши його за посиланням http://jade.tilab.com/ в каталог examples.

4.4 ВИСНОВКИ

У лабораторній роботі були розглянуті переваги мультиагентної мережі групи легких БПЛА в порівнянні з групою одиночних комплексів БПЛА. За рахунок мультиагентної взаємодії в групі, кожен БПЛА-агент здатний автономно коригувати свою задачу при зміні обстановки в навколишньому середовищі або при зміні глобального завдання. Для створення мультиагентної мережі БПЛА наведена нова трирівнева система управління групою. Апаратна реалізація такої системи управління реальна за рахунок впровадження середнього шару – мікрокомп'ютеру, який обмінюється даними з іншими мікрокомп'ютерами , а також коригує завдання автопілоту. Розглянуті приклади двох основних типів завдань для групи БПЛА-завдання моніторингу місцевості і завдання оптимізації польоту БПЛА. Алгоритм моніторингу місцевості показаний на прикладі дослідження екологічної обстановки в акваторії. Показано, що для будь якого візуального дослідження з допомогою групи БПЛА, алгоритм буде відрізнятися тільки в джерелі сигналу, а в апаратній частині літака відмінність полягатиме тільки в типі обладнання.

Навчальне видання

КАСІЛОВ Олег Вікторович

МУЛЬТИАГЕНТНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ В ІГРОВИХ ДОДАТКАХ

Роботу до видання рекомендував *О.А. Сєрков*

Редактор *Л.П. Гобельовська*
Комп'ютерна верстка *Ю.Ф. Власова*
Коректор *Г.В. Сільченко*

Підписано до друку 07.02.2018 р.
Формат 60x84 1/16. Папір офсетний.
Цифровий друк.
Гарнітура DINPro. Ум. друк. арк. 4,88.
Наклад 50 прим. Зам. № ГЛ00-000476.

Видавець і виготовлювач:
ТОВ «Друкарня Мадрид»
61024, м. Харків, вул. Максиміліанівська, 11
Тел.:(057)756-53-25
www.madrid.in.ua info@madrid.in.ua

Свідоцтво суб'єкта видавничої справи:
ДК № 4399 від 27.08.2012 р.


9 786177 683031