Міністерство освіти і науки України
Національний університет «Львівська політехніка»

На правах рукопису

ОБОРСЬКА ОКСАНА ВОЛОДИМИРІВНА

УДК 519.7:004.89

МЕТОДИ ТА ЗАСОБИ МОДЕЛЮВАННЯ ПЕТЛІ БОЙДА У ВІЙСЬКОВИХ ЗАСТОСУВАННЯХ З ВИКОРИСТАННЯМ ОНТОЛОГІЧНОГО ПІДХОДУ

01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем

Дисертація на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник:
Литвин Василь Володимирович
доктор технічних наук, професор,
завідувач кафедри інформаційних систем та мереж
Національного університету «Львівська політехніка»

Львів - 2016

ЗМІСТ

СПИСОК АБРЕВІАТУР ТА ПРИЙНЯТИХ СКОРОЧЕНЬ

OWL – Ontology Web Language;

SWRL – Semantic Web Rule Language;

АСУ – автоматизована система управління;

СВ – Сухопутні війська;

ЗСУ – Збройні Сили України;

ЄАСУ – єдина автоматизована система управління;

ІС – інформаційна система;

БД – база даних;

БЗ – база знань;

ІМ – імітаційне моделювання;

СППР – система підтримки прийняття рішень;

СпППР – підсистема підтримки прийняття рішень;

OODA – петля Бойда (observe, orient, decide and act);

ПО – предметна область;

ППО – протиповітряна оборона;

ОПР – особа, що приймає рішення;

ТЛ – тактична ланка;

ТТХ – тактико-технічні характеристики;

ОВТ – озброєння та військова техніка.

СПИСОК ПОЗНАЧЕНЬ

O – онтологія;

C – множина понять (концептів) онтології;

R – множина відношень між поняттями;

F – скінченна множина функцій інтерпретації;

W – вага важливості поняття;

U – множина військ;

Q – множина військ противника;

M_n – математичне сподівання ураженння цілей;

p – ймовірність ураження цілі.

ВСТУП

Актуальність теми

Ефективність застосування військ (сил) сучасних збройних сил в значній мірі залежить від рівня розвитку системи управління, який, у свою чергу, визначається ступенем їх автоматизації. Автоматизація управління може суттєво підвищити бойові можливості військ (сил) і одночасно в декілька разів скоротити час, які витрачають органи управління на планування дій і доведення завдань до підлеглих. Автоматизована система управління (АСУ) тактичної ланки Сухопутних військ збройних сил України (СВ ЗСУ) – це сукупність взаємозалежних органів та пунктів управління, обладнаних комплексом комп'ютерних апаратно-програмних засобів підтримки прийняття рішень та засобів зв'язку, що забезпечують ефективне управління з'єднаннями, частинами і підрозділами як під час військових зіткнень, так й під час навчання та підготовки військових кадрів. Підсистема підтримки прийняття рішень (СпППР) є центральною компонентою такої АСУ. Вона дає змогу моделювати перебіг бойових дій, виробляти близькі до оптимальних за певними критеріями варіанти рішень та надавати їх для рекомендацій командирам тактичних ланок. Слід зазначити, що середовище в якому функціонує така СпППР, є конкурентним, тобто взаємодіє кілька суб'єктів управління, які є суперниками. Сучасний підхід до моделювання процесу підтримки прийняття рішень у конкурентному середовищі полягає у використанні петлі Бойда, що передбачає багаторазове повторення циклу, який складається з чотирьох послідовних взаємодіючих процесів (етапів): спостереження (observation); орієнтація (orientation); прийняття рішення (decision); дія (action). Таку петлю Бойда в літературі також називають петлею OODA (перші букви назв 4 процесів). Згідно із гіпотезою Бойда – вища швидкість свого циклу і точність оцінок етапів циклу забезпечує перевагу над противником і веде до перемоги у

військових діях.

В Україні та колишньому Радянському Союзі значний внесок у розробленні методів та засобів підтримки прийняття рішень в конкурентному середовищі, зокрема у військовій сфері, зробили такі вчені як А. Я. Вайнер, Є. С. Вентцель, П. Н. Ткаченко, Л. Н. Куцев, Г. А. Мещеряков, А. М. Чавкін, А. Д. Чебикін, І. Я. Дінеру, В. П. Кравченко, Л. А. Овчаров, за кордоном можна відзначити таких вчених як Д. Бойд, Е. Хові, Г. Х. Гуд, Р. Є. Макол.

Під час математичного моделювання бойових дій можна виділити ряд важливих показників, які безпосередньо впливають на їх результат. До таких показників відносяться: відстань між військами; характеристики можливостей військ; прохідність місцевості (коефіцієнт супротиву руху); видимість цілі (ймовірність виявлення цілі); ймовірність знищення цілі; сектор пошуку цілі; щільність розподілу вогневих засобів по цілям противника; число необхідних пострілів для знищення цілі (характеристика розсіювання, захищеність цілі, відстань) тощо. Значення цих показників напряму залежить від бойового статуту, тактико-технічних характеристик (ТТХ) різних видів озброєння та військової техніки (ОВТ), організаційно-штатної структури з’єднань, частин і підрозділів. Окрім того ЗСУ зараз знаходяться на перехідному етапі, а саме вводяться нові стандарти. Отже необхідно враховувати відповідність між новими та старими стандартами. Тому необхідні потужні програмні засоби для зберігання відповідної інформації у базі знань (БЗ). Оскільки ТТХ ОВТ та організаційно-штатна структура військ ґрунтується на певних нормативних документах (бойових статутах), то ядром такої БЗ є онтологія СВ ЗСУ.

Саме тому, розроблення методів та засобів побудови СпППР на основі петлі Бойда з використанням онтологічного підходу є актуальним завданням, а результати таких наукових досліджень надають нові можливості щодо аналізу та синтезу пропонованих рішень у конкурентному середовищі.

Зв’язок роботи з науковими програмами, планами, темами

Дисертаційну роботу виконано в межах наукового напряму «Нові комп’ютерні засоби та технології інформатизації суспільства» визначеного пріоритетним у переліку актуальних проблем Міністерством освіти і науки України, концепції програми інформатизації НАН України, визначеної пріоритетним напрямом, згідно розпорядження № 146 від 27.02.2004 р. та за тематикою наукових досліджень кафедри інформаційних систем і мереж Національного університету «Львівська політехніка», зокрема за темою: «Розроблення інтелектуальних розподілених систем на основі онтологічного підходу з метою інтеграції інформаційних ресурсів», номер державного реєстру 0115U004228 (автор розробив метод підтримки прийняття рішень на основі онтологій в конкурентному середовищі, що дало змогу враховувати специфіку предметної області під час прийняття рішень командирами тактичних ланок).

Мета і завдання дослідження

Метою дисертаційної роботи є розвиток методів підтримки прийняття рішень у конкурентному середовищі на основі петлі Бойда шляхом використання онтологій.

Для досягнення поставленої мети, розв’язано такі *задачі*:

- провести аналіз відомих математичних моделей та методів підтримки прийняття рішень у конкурентному середовищі для формування можливих напрямів їх розвитку;
- розробити модель петлі OODA з використанням онтологічного підходу у вигляді скінченного автомату, який би враховував взаємодію етапів цієї петлі з онтологією;
- розробити методи та алгоритми підтримки прийняття рішень на кожному із етапів петлі OODA з використанням онтології, які б стали основою для програмної реалізації відповідних етапів;

- побудувати архітектуру підсистеми підтримки прийняття рішень, яка враховує запропонований онтологічний підхід, а функціональне наповнення компонент підсистеми відповідає етапам петлі Бойда;
- провести апробацію розроблених методів шляхом реалізації підсистеми підтримки прийняття рішень та проведення експериментальних досліджень.

Об'єктом дослідження є процеси підтримки прийняття рішень у конкурентному середовищі.

Предметом дослідження є методи та засоби побудови систем підтримки прийняття рішень на основі петлі Бойда з використанням онтологічного підходу.

Методи дослідження: для досягнення поставленої мети використано: теорію множин та методи подання знань для моделювання структури онтології та розроблення процедур опрацювання онтологічних знань; теорії дослідження операцій, ймовірностей та штучного інтелекту – для розроблення функціональності окремих модулів СпППР; методи системного аналізу, методи об'єктно-орієнтованого аналізу і проектування – для розроблення архітектури СпППР; теорію реляційних баз даних, методи штучного інтелекту, об'єктно-орієнтоване програмування – для програмної реалізації розроблених моделей, методів та алгоритмів функціонування окремих модулів СпППР.

Наукова новизна одержаних результатів

У дисертаційній роботі вирішено наукове завдання розроблення методів та засобів для побудови систем підтримки прийняття рішень у конкурентному середовищі на основі петлі Бойда з використанням онтологічного підходу.

При цьому отримано такі нові наукові результати:

- вперше розроблено модель петлі OODA з використанням онтологічного підходу, яку подано у вигляді автомата Мура, що дає можливість використовувати онтологічні знання під час етапів петлі OODA, що у свою чергу, дало змогу розробити архітектуру та математичне забезпечення

функціонування СпППР у конкурентному середовищі;

- отримав подальший розвиток метод використання онтологій у конкурентних середовищах, а саме у військових застосуваннях, за рахунок визначення експертами ваг окремих елементів онтології та подання експертних знань за допомогою дескриптивної логіки, що дало змогу підвищити ефективність етапів «Орієнтація» та «Рішення» петлі Бойда під час імітаційного моделювання бойових дій та цілерозподілу, а саме в окремих випадках ймовірність неураження власних військ, отримана модулем імітаційного моделювання з використанням онтології, до 20 % вища у порівнянні з випадком невикористання онтологічних знань;
- удосконалено метод цілерозподілу на основі методів штучного інтелекту, а саме генетичних алгоритмів та опрацювання онтологічних знань, який на відміну від існуючих, не містить процедур повного перебору, що дало змогу зменшити обчислювальну складність алгоритму пошуку ефективного цілерозподілу;
- удосконалено архітектуру СпППР у військових застосуваннях, до складу якої, відповідно до етапів петлі OODA, входять модулі збирання та опрацювання розвідувальних даних, імітаційного моделювання бойових дій, пошуку ефективного цілерозподілу та корегування стрільби, яка містить усі необхідні процеси реалізації петлі Бойда та ґрунтується на онтологічних знаннях, що дало змогу реалізувати СпППР у складі АСУ тактичної ланки СВ ЗСУ.

Практичне значення одержаних результатів

Практичну цінність отриманих наукових результатів дисертаційної роботи полягає у тому, що на основі запропонованих та удосконалених методів підтримки прийняття рішень у конкурентному середовищі на основі петлі Бойда шляхом використання онтологій створено програмний комплекс, який застосовується для підтримки прийняття рішень командирів тактичних ланок. Завдяки розробленому програмному забезпеченню підвищується ефективність

підготовки військових кадрів, зокрема позаштатних, котрі не мають фахових знань. Завдяки автоматизації основних процедур, які реалізовані у програмному комплексі, до 30 % скорочується час на прийняття рішення командиром тактичної ланки.

Результати дисертаційних досліджень у державному підприємстві «Львівський науково-дослідний радіотехнічний інститут» (м. Львів) під час виконання науково-дослідних робіт із створення моделі прийняття рішень в перспективній автоматизованій системі управління тактичної ланки; у військовій частині А2129 Міністерства оборони України (м. Дніпропетровськ) для корегування стрільби артилерії та передачі розвідувальних даних, а також для отримання стійких навичок в процесі підготовки корегувальників артилерії. Використання результатів дисертаційної роботи підтверджено актами про впровадження.

Результати дисертаційної роботи використовуються у навчальному процесі в Національному університеті «Львівська політехніка» у лекційних курсах дисциплін «Теорія прийняття рішень», «Технології підтримки процесів прийняття рішень» при підготовці студентів за спеціальністю «Системи і методи прийняття рішень», що підтверджено відповідними актами.

Апробація результатів дисертації

Основні наукові та практичні результати роботи оприлюднено та обговорено на: Міжнародній науково-технічній конференції «Перспективи розвитку озброєння та військової техніки Сухопутних військ» (м. Львів, 2015 р.); IV-й Міжнародній науковій конференції студентів, аспірантів та молодих вчених «Теоретичні та прикладні аспекти кібернетики» ТААС-2014 (м. Київ, 2014 р.); XIIIth International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» (м. Свалява, 2015 р.); 9th International Scientific and Technical Conference «Computer Sciences and Information Technologies» CSIT-2014 (м. Львів, 2014 р.); X-й Міжнародній науково-практичній конференції «Математичне та імітаційне моделювання

систем» МОДС-2015 (м. Чернігів, 2015 р.); II-й, III-й та IV-й Міжнародних наукових конференціях «Інформація, комунікація, суспільство» ІКС-2013, ІКС-2014, ІКС-2015 (м. Славське, Львівська область, 2013, 2014 та 2015 рр.); Міжнародній науково-практичній конференції «Математика. Інформаційні технології. Освіта» (м. Луцьк, 2013 р.); Міжнародних науково-практичних конференціях «Математика. Інформаційні технології. Освіта» (м. Луцьк, 2013, 2014 рр.); III-й Науково-технічній конференції Фізико-механічний інститут ім. Г. В. Карпенка НАН України «Обчислювальні методи і системи перетворення інформації» (м. Львів, 2014 р.); Міжнародній науковій конференції «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту» (м. Херсон, 2014 рік); III-й Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених «Математичні методи, моделі та інформаційні технології» YESS-2014 (м. Чернівці, 2014 р.).

Матеріали досліджень апробовані на трьох локальних кафедральних наукових семінарах у повному обсязі.

Публікації

За результатами дисертаційних досліджень опубліковано 19 наукових праць, серед яких: 1 стаття у закордонному періодичному виданні, 5 статей у наукових виданнях України, 13 публікацій у матеріалах наукових конференцій.

Короткий зміст дисертації

Матеріал дисертаційної роботи розподілений таким чином.

У **вступі** обгрунтовано актуальність теми, сформульовано мету та основні завдання досліджень, показано зв’язок із науковими програмами, планами, темами, розкрито наукову новизну. Розглянуто практичну цінність, реалізацію та впровадження результатів роботи. Наведено дані про особистий внесок здобувача, апробацію роботи та публікації.

В **першому розділі** здійснено аналіз методів моделювання у військових застосуваннях з використанням онтологічного підходу. За основу для

моделювання обрано кібернетичну модель OODA (Observation-Orientation-Decision-Action), запропоновану Джоном Бойдом. Така модель передбачає багаторазове повторення петлі, яка складаються з чотирьох послідовних взаємодіючих процесів: спостереження (observation); орієнтація (orientation); прийняття рішення (decision); дія (action). На основі такої моделі розроблено архітектуру підсистеми підтримки прийняття рішень (СпППР), основною компонентою якої є база знань (БЗ) [128]. Запропонована СпППР є центральною компонентою автоматизованої системи управління (АСУ) тактичною ланкою Сухопутних військ збройних сил України (СВ ЗСУ). Знання, які використовуються в цій предметній області є об'єктивними й містяться у нормативних документах (військовий статут, тактико-технічні показники, нормативні показники тощо). Тому запропоновано в якості ядра БЗ такої СпППР використати онтологію.

Надалі постають такі задачі як розроблення методів та засобів використання онтології в петлі OODA, побудова онтології СВ ЗСУ, розроблення програмних модулів у складі СпППР моделювання поведінки інтелектуального агента (тактичної ланки) в конкурентному середовищі на основі онтологій. Розв'язування цих задач наведено у наступних розділах роботи.

В **другому розділі** розроблено метод використання онтологій в петлі OODA. Розроблено математичне забезпечення кожного з етапів петлі.

Згідно із робленим методом, зміст онтології напряму впливає на 2-й і 3-й етапи петлі OODA, а сама структура та наповнення онтології залежить від 1-го та 2-го етапів.

В **третьому розділі** розроблено онтологію СВ ЗСУ, правила поведінки в певних ситуаціях на основі дескриптивної логіки та алгоритмів функціонування СпППР на різних етапах петлі OODA.

Також у цьому розділі розроблено алгоритми функціонування модулів розвідування, імітаційного моделювання та цілерозподілу на основі

генетичного алгоритму, які входять у склад СпППР. Розроблено архітектуру АСУ СВ ЗСУ, центральною компонентою якої є побудована СпППР.

В **четвертому розділі** наведено результати практичного впровадження запропонованих методів та засобів. У склад СпППР входять такі модулі: збору, обробки та передачі розвідувальних даних (мобільний застосунок «Military intelligence»), імітаційного моделювання перебігу бойових дій, цілерозподілу, корегування стрільби (мобільний застосунок «Adjustment»).

За допомогою розроблених методів і програмних засобів, експериментально доведено, ефективність розробленої СпППР, що дала змогу до 30 % скоротити час, який витрачають командири тактичних ланок на планування і доведення завдань до підлеглих; покращення тактичної і бойової підготовки військових кадрів СВ ЗСУ.

У **висновках** наведено основні результати виконаної роботи.

У **додатках** наведено фрагменти програмних кодів модулів розробленої СпППР, часткова онтологія СВ ЗСУ, а також акти впровадження результатів дисертаційної роботи.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ МОДЕЛЮВАННЯ ПРОЦЕСУ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ЗА ДОПОМОГОЮ ПЕТЛІ БОЙДА

У першому розділі здійснено аналіз методів моделювання у військових застосуваннях з використанням онтологічного підходу. Для моделювання такого процесу обрано кібернетичну модель Джона Бойда (OODA). Здійснено аналіз методів реалізації такої моделі. Запропоновано використати онтологію предметної області в якості основної компоненти під час моделювання процесу підтримки прийняття рішень у військових застосуваннях.

1.1. Загальні принципи побудови АСУ командирами тактичних ланок СВ ЗСУ

Добре організоване і безупинне управління підрозділами в бою забезпечує захоплення й утримання ініціативи, скритність підготовки і раптовість нанесення ударів по противнику, ефективне використання наявних засобів ураження та повне використання бойових можливостей частин і підрозділів.

Сьогодні в арміях провідних країн світу першочергова увага приділяється підвищенню ефективності управління бойовими діями.

Одним із шляхів досягнення цієї мети є забезпечення всеохоплюючої інформаційної переваги над противником на базі глобальної ситуаційної обізнаності командирів і штабів в реальному масштабі часу. Інформація про обстановку на полі бою стає основою для інтеграції різних автоматизованих систем, що дозволяє прийняти раціональне рішення та досягти максимального ефекту застосування зброї. В основу інформаційної інфраструктури збройних

сил покладається сукупність інформаційних систем різного рівня управління, взаємопов’язаних як по вертикалі, так і по горизонталі.

Основними тенденціями розвитку (АСУ) військами збройних сил провідних країн світу є [80, 118-120]:

- стандартизація обладнання у ланках управління «батальйон»,
- «бригада», «армійський корпус»;
- побудова на основі відкритих архітектур, які забезпечують простоту взаємодії з уніфікованими інтерфейсами і їх еволюційну модернізацію з метою покращення показників роботи та розширення функціональних можливостей;
- інтеграція із системами управління видів збройних сил, родів військ, об’єднаних сил, системами союзників;
- гнучкість застосування, коли командири мають можливість групувати модулі у різних конфігураціях, а обладнання є виносним;
- реалізація концепції розподілених систем, коли обладнання розноситься на відстань від декількох десятків до декількох сотень метрів в одній загальній зоні, але працює як одне ціле;
- забезпечення дублювання виконуваних функцій для реалізації систем, що розділяються, відповідно до яких модулі «скороченого» центра продовжують виконувати всі функції повного складу під час його передислокації по складених компонентах, хоча й при меншому обсязі оброблюваної інформації;
- забезпечення зв’язку між елементами системи у будь-якій конфігурації (зосередженої або розподіленої);
- надання можливості тактичної телеконференції;
- наявність надлишкових можливостей для маршрутизації інформації в межах системи;
- наявність засобів відображення, що забезпечують колективну роботу посадових осіб;

- забезпечення гарантованого зберігання секретної інформації та документації.

Інша ситуація склалася у Збройних Сил України (ЗСУ). У даний час АСУ Сухопутних військ (СВ) – відсутня. Однак позитивним є усвідомлення військовим керівництвом у всіх ланках управління необхідності якнайшвидшої її розробки та впровадження як складової Єдиної автоматизованої системи управління (ЄАСУ) ЗСУ. Науковці Наукового центру Сухопутних військ цілеспрямовано працюють над обґрунтуванням загальних принципів побудови, точнісних критеріїв ефективності навігаційних засобів при виконанні завдань за призначенням частинами і підрозділами, часових критеріїв передачі інформації, яка циркулює в АСУ, розробкою алгоритмів роботи посадових осіб пунктів управління тактичної ланки (ТЛ) СВ ЗСУ при підготовці і веденні бойових дій, методики побудови та інфологічної моделі бази даних АСУ ТЛ [121].

Вироблені такі погляди на принципи побудови АСУ ТЛ СВ ЗСУ. Щодо визначення поняття «автоматизована система управління підрозділу» немає загальноприйнятого погляду. Пропонується наступне визначення: автоматизована система управління підрозділу – це система взаємозалежних органів управління, командно-спостережних пунктів та комплексу комп’ютерних апаратно-програмних засобів підтримки прийняття рішень та засобів зв’язку, що забезпечують ефективне управління підрозділом.

Основними воєнно-технічними проблемами проектування АСУ є:

- вибір інформаційної моделі;
- визначення складу, призначення і порядку взаємодії її елементів;
- раціональний розподіл функцій між посадовими особами і засобами автоматизації;
- організація взаємодії посадових осіб із засобами автоматизації;
- алгоритмізація задач управління, рішення яких покладається на АСУ;
- автоматизація процесів отримання, обробки і передачі інформації.

Існують дві інформаційні моделі АСУ військами. Відповідно до першої, ієрархічної інформаційної моделі, АСУ повинна відповідати організаційній структурі військ та повторювати ієрархію органів управління. Єдиною її перевагою є простота. Недоліки набагато численніші і суттєвіші. По-перше, в умовах постійного реформування організаційної структури, органів управління спроектувати та побудувати АСУ за такою моделлю неможливо. По-друге, копіювання в структурі АСУ ієрархії органів управління не дасть значного підвищення оперативності, тому що інформацію органи управління отримують послідовно; стійкість та безперервність взагалі залишаться на тому ж рівні, оскільки ураження хоча б одної проміжної ланки призводить до порушення управління в цілому напрямку. По-третє, кожний орган управління отримує інформацію не від самих об’єктів управління, а з доповідей органу управління нижчої ланки; при цьому можливе умисне або неумисне спотворення інформації. По-четверте, інформація між взаємодіючими органами управління, як правило, передається через органи управління вищих ланок і формується із доповідей, що негативно впливає на оперативність і достовірність.

Другою є модель незалежного управління (мережецентрична), в якій, на противагу ієрархічній, підпорядковані та взаємодіючі органи управління отримують інформацію від об’єктів управління паралельно і незалежно, обробляють її та приймають рішення в межах своєї відповідальності; інформація про обстановку від нижчих органів управління передається одразу після змін, а не з визначеною періодичністю; алгоритми роботи командира і штабу при цьому не змінюються. Тому така інформаційна модель АСУ військами є незалежною від ієрархії органів управління та структури збройних сил, що постійно змінюються, і дозволяє суттєво підвищити оперативність, стійкість та безперервність управління, достовірність отримуваної інформації та, відповідно, адекватність рішень, що приймаються. Недоліками такої моделі є відносна складність, підвищені вимоги до засобів обробки та передачі інформації і, як наслідок, висока вартість при технічній реалізації. Однак

сучасні досягнення інформаційних та телекомунікаційних технологій дозволяють реалізувати модель незалежного управління.

Вирішуючи проблему розподілу функцій управління між особами командного пункту (штабу) і засобами автоматизації, необхідно виходити з того, що засоби автоматизації повинні слугувати робочим інструментом командира, осіб командного пункту (штабу), забезпечувати їх вичерпною інформацією про обстановку, дозволяючи їм своєчасно прийняти обґрунтовані рішення.

При організації взаємодії осіб командного пункту (штабу) із засобами автоматизації необхідно враховувати, що із комплексу задач обробки інформації і управління в першу чергу підлягають автоматизації задачі, що мають масовий характер, піддаються формалізації і вимагають виконання великого числа обчислювальних і логічних операцій за обмежений час. При визначенні доцільного ступеня автоматизації рішення тих чи інших задач враховується відносна ефективність їх вирішення людиною та електронно-обчислювальною машиною, економічні витрати, наявний запас часу.

Основними функціональними складовими АСУ ТЛ є [152]:

- підсистема інформаційного обміну (телекомунікаційна);
- підсистема оброблення даних;
- підсистема збору та первинного оброблення інформації;
- підсистема інформаційно-аналітичної обробки даних та підтримки прийняття рішень;
- підсистема реалізації управлінських рішень;
- підсистема забезпечення безпеки інформації;
- підсистема управління функціонуванням;
- підсистема забезпечення;
- підсистема науково-технічного супроводження.

Функціональна підсистема інформаційного обміну (телекомунікаційна) призначена для формування, передачі і прийому бойових документів,

забезпечення мовного зв’язку та оперативного обміну користувачів повідомленнями, доведення команд по управлінню діями підрозділів, своєчасної і достовірної передачі інформації між пунктами управління, забезпечення одночасного сумісного спілкування між окремими користувачами та групами розосереджених користувачів АСУ на основі передачі й оброблення мовної, текстової та відеоінформації.

Функціональна підсистема оброблення даних призначена для оброблення, збереження та надання доступу до даних (інформації), які обробляються (використовуються) в ситуаційних центрах. Вона утворена потужними обчислювальними серверами та засобами збереження (архівування) даних.

Функціональна підсистема збору та первинного оброблення інформації призначена для формування, зберігання та надання посадовим особам графічної та текстової інформації, необхідної для прийняття рішень при підготовці та веденні бойових дій, забезпечення користувачів довідковою інформацією для виконання їх функціональних обов’язків, автоматизації діяльності з діловодства, розробки та контролю виконання документів, ведення баз даних про стан власних підрозділів, поточну обстановку, підрозділи противника, систематизації розвідувальної інформації, організації зручного доступу до баз даних, розмежування доступу. Утворена засобами моніторингу, збору та оброблення різнорідної інформації.

Функціональна підсистема інформаційно-аналітичної обробки даних та підтримки прийняття рішень призначена для проведення оперативно-тактичних розрахунків, що забезпечують підтримку прийняття рішень посадових осіб всіх ланок, визначення механізмів збору та обробки інформації, а також засобів її аналізу, систематизації та класифікації, моделювання бойових дій. можливість індивідуального та колективного прийняття рішень на будь-якому етапі його ухвалення. Утворена переважно програмно-технічними засобами оброблення інформації (з відповідним забезпеченням) і засобами відображення (візуалізації) даних, передбачає використання спеціальних засобів зв’язку та

передачі даних для забезпечення колективного прийняття рішень (наприклад, відеоконференцзв’язок).

Функціональна підсистема реалізації управлінських рішень призначена для забезпечення та реалізації безпосереднього управління (керівництва) підпорядкованими об’єктами (силами, засобами) при виконанні ними поставлених завдань (реалізації планів управління). Вона забезпечує формування, передачу і прийом керівних (розпорядчих) документів (наказів, розпоряджень, доповідей, команд, сигналів управління, фото- та відеоідокументів тощо), команд та сигналів управління, мовний зв’язок та обмін короткими текстовими повідомленнями між посадовими особами. Утворена програмно-технічними засобами оброблення даних (з відповідним забезпеченням) для проведення необхідних оперативних розрахунків, передбачає використання засобів зв’язку та передачі даних, до яких ставляться підвищені вимоги щодо оперативності та достовірності.

Функціональна підсистема забезпечення безпеки інформації призначена для захисту інформації, яка циркулює у всіх ланках, від несанкціонованого доступу, ідентифікації та аутентифікації користувачів. Утворена блоками (елементами) функціонального контролю, вбудованими у визначений перелік модулів ситуаційних центрів, засобами збору та оброблення інформації, організаційно-методичним забезпеченням (організаційно-методичною та технічною документацією) конфігурування системи.

Функціональна підсистема управління функціонуванням призначена для контролю працездатності та налаштування (конфігурування) всіх складових елементів (модулів) ситуаційних центрів. Утворена блоками (елементами) функціонального контролю, вбудованими у визначений перелік модулів ситуаційних центрів, засобами збору та оброблення інформації, організаційно-методичним забезпеченням (організаційно-методичною та технічною документацією) конфігурування системи.

Функціональна підсистема забезпечення виконує внутрішні функції забезпечення, такі як: енергоживлення, створення нормальних умов життєдіяльності особового складу, належних умов експлуатації апаратури тощо. Утворена засобами відповідних систем забезпечення.

Функціональна підсистема науково-технічного супроводження забезпечує обґрунтування та проведення заходів цілеспрямованого своєчасного коригування дій користувачів засобів ситуаційних центрів щодо підвищення ефективності їх використання та освоєння нововведень, виявлення проблем у функціональних системах (в усіх режимах роботи).

Таким чином, АСУ відіграють вирішальну роль в забезпеченні ефективного управління військами та зброєю; враховуючи основні світові тенденції, найбільш прийнятним шляхом є розроблення АСУ ТЛ за мережецентричним принципом з поступовим нарощуванням її можливостей; розробка і впровадження АСУ ТЛ СВ ЗСУ є складною комплексною науково-технічною проблемою [89].

Центральною компонентою такої АСУ є СпППР командирами ТЛ. Основною властивістю такої СпППР є те, що вона функціонує у конкурентному середовищі. Конкурентне середовище – це результат і умови взаємодії великої кількості суб’єктів управління, що визначає відповідний рівень суперництва і його вплив на процес прийняття рішень.

1.2. Підходи до моделювання процесу підтримки прийняття рішень у конкурентному середовищі

Відповідно з ідеями Джона Бойда та його послідовників будь-яка діяльність у конкурентному середовищі з певним ступенем наближення може бути представлена у вигляді кібернетичної моделі OODA [3] (рис. 1.1), яка передбачає багаторазове повторення циклу, який складається з чотирьох послідовних взаємодіючих процесів: спостереження (observation); орієнтація (orientation); прийняття рішення (decision); дія (action).

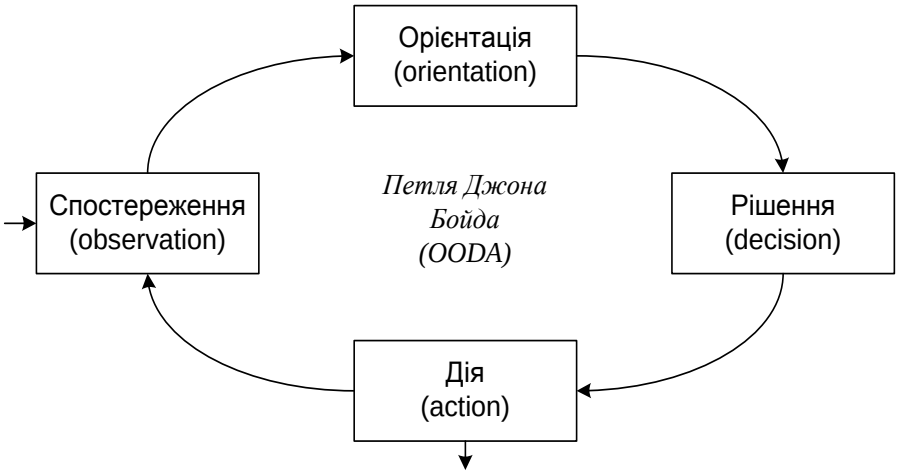

Рис. 1.1. Процеси петлі OODA

Така модель з успіхом почала застосовуватися для моделювання діяльності та прийняття рішень у бізнесі, політиці, соціології тощо, тобто у тих сферах, де наявна конкуруюча сторона [47].

Згідно теорії Бойда кожна людина, або організація при вирішенні поставлених перед ними завдань має свою петлю прийняття рішень і діяльності.

Петля OODA розглядається як єдина типова модель циклу прийняття рішень для систем командування та управління, як своїх військ, так й військ противника. На думку багатьох вчених, теорія Джона Бойда застосовується в різноманітних галузях наукового та практичного знання [7-12, 96]. Більшість авторів вказаних робіт констатують, що цикл OODA за складом функціональних блоків, модельними та когнітивними можливостями є певною «золотою серединою» серед усіх найбільш відомих циклічних моделей, які використовуються у процесі прийняття рішень та у воєнній справі. Необхідно зазначити відповідність петлі Джона Бойда загальній методології наукового методу: спостереження – формування гіпотези – перевірка гіпотези – побудова теорії, яка відповідає даним спостереження. Відмінна риса циклу OODA від інших циклічних моделей полягає в тому, що в будь-якій ситуації завжди передбачається наявність противника, з яким ведеться озброєна боротьба. Противник також діє та приймає рішення в межах своєї аналогічної петлі.

На рис. 1.2 представлена модель збройної боротьби з урахуванням циклу OODA двох супротивних сторін [1].

1.3. Аналіз етапів петлі OODA

Людська дія відбувається через взаємодію з п’яти чинників: подій, сприйняття, порівняння, пізнання, і відповідей. Здебільшого, поведінка складається з зовнішньої події і чотирьох внутрішніх завдань. Розуміння цього призвело Джона Бойда розробити модель в якій говорилося, що поведінка людини може бути описана як безперервний, інтерактивний процес який складається з чотирьох етапів: спостереження, орієнтації, прийняття рішення та дія, широко відомого як цикл OODA [1-4].

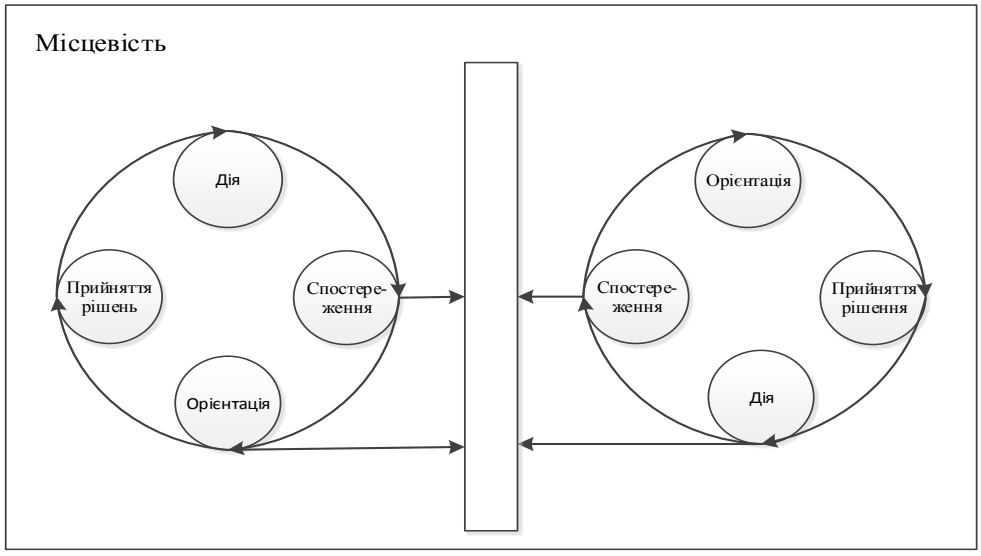

Рис. 1.2. Модель збройної боротьби з урахуванням циклів OODA двох протиборчих сторін

Фактично має місце розвиток ситуації по спіралі і на кожному етапі цієї спіралі здійснюється взаємодія з зовнішнім середовищем та вплив на противника. Модель зазвичай відносять до розряду кібернетичних, так як в ній реалізується принцип «зворотного зв’язку», відповідно до якого частина виходу з системи знову подається на її вхід, щоб уточнити, а якщо буде потрібно, і скоригувати розвиток системи на наступних етапах. Ця модель з успіхом

почала застосовуватися для моделювання діяльності та прийняття рішень у бізнесі, політиці та соціології.

За теорією Бойда кожна людина, або організація при вирішенні поставлених перед ними завдань має свою петлю ухвалення рішень і діяльності. Розглянемо більш детально кожен з чотирьох окремих елементів зазначеної петлі.

1. Спостереження (observation) – це процес збору інформації, необхідної для прийняття рішення в даному конкретному випадку. Необхідна інформація може бути отримана як від зовнішніх, так і від внутрішніх джерел. Під внутрішніми джерелами інформації розуміються елементи зворотного зв'язку петлі. В якості зовнішніх використовуються датчики, а також інші канали отримання інформації.

Щоб спостереження набуло наукового характеру, потрібно, аби воно:

1) було планомірним, а не випадковим;
2) здійснювалося послідовно й систематично;
3) було забезпечене достатньо широкою інформацією про явище, яке є предметом спостереження (слід оперувати якомога більшою кількістю фактів);
4) передбачало точну фіксацію результатів спостереження.

Спостереження висуває вимоги й до особистих якостей дослідника. Він повинен:

- бути об'єктивним при фіксації, усному описі та класифікації фактів спостереження;
- володіти собою, щоб його власні почування та особисті характерологічні якості не впливали на його роботу, не позначалися на спостереженні та не спотворювали висновків;
- не бути тенденційним, упередженим в організації спостереження та очікуванні його наслідків, щоб не дійти безпідставних висновків;
- не піддаватися першим враженням від досліджуваного;

- не бути поблажливим щодо досліджуваного;
- не приписувати досліджуваному своїх власних якостей і не пояснювати його поведінку з власних позицій.

Об’єктивності потрібно дотримуватись протягом усього процесу дослідження. Об’єктивність має бути визначальним чинником вірогідних висновків.

Найбільш ефективним і плідним з наукового погляду є експериментальне дослідження, особливість якого полягає в тому, що явище (предмет дослідження) вивчається за різних умов та обставин. Застосування цього методу дослідження сприяє глибокому і дуже точному вивченню певної психологічної закономірності.

2. Орієнтація (orientation) – найбільш відповідальний і найбільш складний з когнітивної точки зору етап у всьому циклі OODA [88]. Етап орієнтації складається з двох підетапів: руйнування (destruction) і творення (creation). Руйнування передбачає розбиття ситуації на дрібні елементарні частини, які більш легкі для розуміння. Людина або організація, які приймають рішення, будуть прагнути розчленувати, або декомпонування завдання до такого рівня, поки новоутворені складові завдання стають близькими до стандартних, або типових ситуацій, для яких особа, що приймає рішення (ОПР) має план дій. Ознайомлення з цими елементарними типовими підзадачами досягається шляхом навчання, тренування, накопичення досвіду та інструктажу. Таке ознайомлення можливо тільки на основі заздалегідь розроблених доктринальних установок і безлічі планів. ОПР ідентифікує поточну ситуацію по відношенню з тими, з якими вона знайома, і застосовує заздалегідь заготовлений план дій для цієї підзадачі. Потім ці складові елементарні підплани об’єднуються в загальний план дій, що і відповідає підетапу «творення». Якщо немає реальних планів, з числа яких може бути обрано рішення, то процес залишається на етапі орієнтації і здійснюється подальша

декомпозиція задачі. Якщо не вдається розробити план з реальними шансами на успіх, то подальше подрібнення може призвести до останнього циклу.

Для етапу орієнтації використовуються методи аналізу і синтезу які тісним чином пов’язані між собою. Вони призначені для обробки інформації, отриманої в результаті застосування дослідницьких методів.

Аналіз являє собою вивчення якостей, властивостей і характеристик досліджуваного об’єкта за допомогою його умовного поділу на окремі складові частини. У свою чергу синтез полягає в узагальненні інформації про окремих складових і формуванні сукупності інформаційних даних про об’єкт дослідження в цілому.

Результати, отримані в процесі аналізу та синтезу, служать основою для складання різного роду прогнозів на найближчу і далеку перспективу. Прогнозування може здійснюватися методами розрахунку і екстраполяції.

3. Прийняття рішення (decision) – третій етап циклу OODA. Якщо до цього етапу ОПР змогла сформувати тільки один реальний план, то приймається рішення – виконувати цей план чи ні. Якщо ж сформовані кілька альтернативних варіантів дій, то ОПР на даному етапі здійснює вибір найкращого з них для подальшої реалізації. Вибір найкращого плану може здійснюватися за критерієм ефективність-вартість. В умовах ліміту часу кращим вважається той план, що відповідає вимогам швидкої надійності.

Для прийняття рішення використовуються такі методи:

- метод ефективність-вартість враховує три етапи: побудова моделі ефективності, побудова моделі вартості, синтез вартості й ефективності;
- метод теорії надійності – властивість технічних об’єктів зберігати у часі у встановлених межах значення всіх параметрів, необхідних для виконання технічних функцій в заданих режимах і умовах застосування.

4. Дія (action) – заключний етап циклу, що передбачає практичну реалізацію обраного курсу дій або плану. Дія передбачає видачу наказу, або вказівки, фізичну атаку, активний захист, переміщення в просторі, або

управління датчиками з метою поліпшення спостереження в наступному бойовому циклі.

Єдиний спосіб покращити вашу петлю OODA – через підготовку. Вправи з усунення затримки є відмінним прикладом. Єдиним способом поліпшити час реагування є тренування. Чим більше часу потрібно вам, щоб увійти в дію, тим більше часу ви надаєте противнику.

1.4. Підходи досягнення переваги в петлі OODA

Існують два основні способи досягнення переваги в різних видах військової діяльності.

Перший спосіб – зробити в якісному вимірі свої цикли дій більш швидкими. Це дозволить діяти першими та змусить противника реагувати на наші дії.

Другий спосіб – покращити якість рішень, які нами приймаються, тобто приймати рішення, які більшою мірою відповідають обстановці, що склалася, ніж рішення противника. Більш якісні рішення можуть привести до кращих результатів, ніж швидкі, зазвичай неадекватні, або погано прораховані дії. Враховуючи вищенаведені міркування, на кожному кроці процесу необхідно прагнути до поступового отримання якісних та кількісних покращень. Розглянемо більш детально ці два способи отримання переваги. Прискорення циклу OODA. Відповідно до теорії Джона Бойда необхідно «регулювати з середини» процес діяльності противника, або його випередити за рахунок більш швидкої власної петлі дій. У свою чергу, прискорення процесу прийняття рішень може мати два ефекти. Перший ефект за своєю природою має виключно наступальний характер. Можна почати впроваджувати свій план першим і тим самим викликати зміни в обстановці перш, ніж почне діяти противник. Якщо для виконання плану необхідна участь противника (наприклад, противник планує зайняти певні рубежі), ініціатива в діях дозволяє нам зайняти ці рубежі перед початком реалізації запланованих дій. Ця перевага першого удару

знаходить своє втілення в простій формулі – ви можете вбити противника перед тим, як він почне стріляти.

Другий ефект від прискорення власного циклу дій OODA носить оборонний характер. Протидіюча сторона з перевагою у швидкості циклу дій спроможна в деяких випадках уникнути вразливої, або шкідливої дії з боку противника. Іншими словами, можна діяти всупереч очікуванням атакуючого противника. Якісне покращення циклу OODA в цьому випадку означає, що якість рішень, які приймаються, буде кращою, ніж у противника. Оцінка рівня якості рішень, які приймаються, є величиною не абсолютною, а відносною, тому досягти конкурентної переваги в цьому компоненті можна двома способами: удосконаленням своїх рішень та погіршенням рішень, які приймає сторона противника.

Підвищення якості власних рішень може бути досягнуто різними способами, до яких відносяться, зокрема, застосування сучасних формальних математичних методів, удосконалення інформаційно-аналітичного та розвідувального забезпечення, застосування автоматизованих систем управління, систем підтримки рішень, експертних та відповідних систем, навчань та тренувань. Навчання та тренування є найбільш загальним способом покращення процесу наших рішень. Саме в ході навчань і тренувань відпрацьовується інформація необхідна в конкретних умовах. Удосконалюючи свій цикл OODA, слід постійно пам’ятати про те, що існують реальні можливості знизити якість циклу прийняття рішень й діяльності противника, шляхом створення завад та протидії системам розвідки й спостереження, введенням противника в оману (на етапах спостереження та оцінки), прийняття нехарактерних та непередбачуваних рішень, які називають сюрпризами (на етапі оцінки).

Послабити ефективність дій противника можна й на етапі застосування зброї (на етапі дії), шляхом використання елементів активного захисту, наприклад теплових і радіолокаційних хибних цілей. Досвід сучасних

конфліктів свідчить про те, що максимальні часові затрати на реалізацію різних складових циклу OODA стали значно меншими у порівнянні з військовими конфліктами попередніх часів. У війнах майбутнього ці часові інтервали ще більше скоротяться, що буде досягнуто за рахунок застосування перспективних датчиків різноманітної природи, інформаційних технологій, технологій телекомунікацій, засобів обчислювальної техніки й швидкодіючих пристроїв.

Найбільш складним етапом, з точки зору скорочення часу, є етап дії, або застосування зброї. Найбільш радикальними способами збільшення швидкості вражаючого фактору до цілі є розвиток гіперзвукових технологій доставки та створення нетрадиційних видів зброї (кінетичної, лазерної та надвисокочастотної). Ключовим моментом для отримання переваги над противником є випередження його у циклі прийняття рішень та дій. Реалізацією максимальної швидкості циклу Джона Бойда є повна автоматизація та інформатизація всіх етапів аналізу та прийняття рішень. Однією із сучасних прогресивних концепцій ведення війни в умовах сучасності є концепція мережецентричної війни (тобто центрованої на мережах, на об’єднанні всіх підсистем в єдину мережу). В умовах будь-якої мережецентричної організації збільшення числа вузлів мережі (у нашому випадку джерел розвідки) та числа зв’язків між ними – ключовий параметр ефективності етапів спостереження (розвідки) та орієнтації (оцінки). Таким чином, відповідно до принципів мережевої війни всі види розвідки (джерела отримання інформації) пов’язують в одну мережу.

Відмітна риса циклу OODA від інших циклічних моделей полягає в тому, що в будь-якій ситуації завжди передбачається наявність конкурентної сторони.

Під час математичного моделювання бойових дій [1-3] можна виділити ряд важливих показників, які безпосередньо впливають на їх результат. До таких показників для моделювання бойових дій СВ відносяться: відстань між військами; характеристики маневрових можливостей військ; прохідність місцевості (коефіцієнт супротиву руху); видимість цілі (ймовірність виявлення

цілі); ймовірність знищення цілі; сектор пошуку цілі; щільність розподілу вогневих засобів по цілям противника; число необхідних пострілів для знищення цілі (характеристика розсіювання, захищеність цілі, відстань тощо).

У більшості випадків значення цих показників напряму залежить від тактико-технічних характеристик (ТТХ) різних видів озброєння та військової техніки (ОВТ) та організаційно-штатної структури з'єднань, частин і підрозділів. Тому необхідні потужні програмні засоби для зберігання відповідної інформації. Така інформація повинна зберігатися в базі знань, а не в базі даних, оскільки під час моделювання бойових дій важливу роль відіграє логічне виведення, яке можна реалізувати на основі знань про предметну область (ПО).

Оскільки ТТХ ОВТ та організаційно-штатна структура військ ґрунтується на певних нормативних документах, то ядром такої БЗ служитиме онтологія СВ ЗСУ. Отже виникає актуальна науково-технічна проблема побудови СпППР, центральною компонентою якої є вище визначена БЗ [4, 110, 112].

Тому пропонується для моделювання петлі OODA використовувати інтелектуальну систему, ядром БЗ якої є онтологія середовища. На нашу думку, зміст онтології напряму впливає на 2-й і 3-й етапи циклу, а сама структура та наповнення онтології залежить від 1-го та 2-го етапів.

1.5. Використання онтологій в системах підтримки прийняття рішень

На сьогодні актуальною є задача формування концептуальних «прозорих» подань для слабо-структурованих предметних областей. Провідною парадигмою структурування інформаційних потоків є онтології, або ієрархічні концептуальні структури, які формуються аналітиком на основі вивчення і структурування потоків інформації, документів, протоколів витягнутих знань та інших джерел [107-109].

Онтологічний інжиніринг (ОІ) розвиває основні положення інженерії знань – науки про моделі і методи добування, структуризації та формалізації знань.

Інженерія знань – це область штучного інтелекту, в той час як ОІ охоплює більш широке коло додатків – від систем управління знаннями до дистанційного навчання [56, 57].

Онтологічний інжиніринг робить перші кроки, тому кожен аналітик йде методом проб і помилок, створюючи складні онтологічні структури, що відображають лабіринти професійних знань у різних прикладних областях.

Онтологія (від грец. онтос – суще, логос – навчання, поняття) – термін, що визначає вчення про буття, про сутність, на відміну від гносеології – вчення про пізнання [94, 95]. Вже у Х. Вольфа (1679-1754), автора терміну «онтологія», вчення про буття було відокремлене від вчення про пізнання. До цього онтологія була частиною метафізики, наукою самостійною, незалежною і не пов’язаною з логікою, з «практичною філософією», з науками про природу. Її предмет складає вивчення абстрактних і загальних філософських категорій, таких як буття, субстанція, причина, дія, явище тощо, а сама онтологія як наука домагалася повного пояснення причин усіх явищ [3, 31, 78-83].

З точки зору, ближчої до понять, пов’язаних зі штучним інтелектом, онтологія – це знання, формально відображені на базі концептуалізації. Концептуалізація – опис множини об’єктів і понять, знань про них і зв’язків між ними. Онтологією називається експліцитна специфікація концептуалізації. Формально онтологія складається з термінів (понять, концептів), організованих в таксономію, їх визначень і атрибутів, а також пов’язаних з ними аксіом і правил виведення.

Забезпечення можливості використання знань предметної області стало однією з рушійних сил недавнього сплеску у вивченні онтологій. Наприклад, для моделей багатьох різних предметних областях необхідно сформулювати поняття часу. Це термін включає поняття тимчасових інтервалів, моментів часу

і т. ін. Якщо одна група вчених детально розробить таку онтологію, то інші можуть просто повторно використовувати її у своїх ПО. Крім того, якщо нам треба створити велику онтологію, ми можемо інтегрувати дещо з існуючих онтологій, які описують частини великої ПО. Ми також можемо повторно використовувати основну онтологію і розширити її для опису ПО, яка нас цікавить.

Створення явних допущень у предметній області, що лежать в основі реалізації, дає можливість легко змінити ці припущення при зміні наших знань про ПО. Жорстке кодування припущень про світ на мові програмування призводить до того, що ці припущення не тільки складно знайти і зрозуміти, але і також складно змінити, особливо не програмісту. Крім того, явні специфікації знань ПО корисні для нових користувачів, які мають розуміти значення термінів ПО [4].

Відокремлення знань предметної області від оперативних знань – це ще один варіант загального застосування онтологій. Ми можемо описати задачу конфігурації продукту з його компонентів відповідно до необхідної специфікації і впровадити програму, яка робить цю конфігурацію незалежною від продукту і самих компонентів. Після цього ми можемо розробити онтологію компонентів і характеристик комп'ютерних комплексів і застосувати цей алгоритм для конфігурації нестандартних комп'ютерних комплексів. Ми також можемо використовувати такий самий алгоритм для конфігурації ліфтів, якщо ми надамо йому онтологію компонентів ліфта [5, 6].

Аналіз знань у предметній області можливий, коли є декларативна специфікація термінів. Формальний аналіз термінів надзвичайно цінний як при спробі повторного використовування існуючих онтологій, так і при їх розширенні.

Знання, описані в онтології, можна використовувати в інших програмах, базах даних і т.д [90, 91, 113-117]. При цьому значно підвищується

ефективність як інтелектуальних систем, так і традиційних інформаційних систем. Цим визначається актуальність створення онтологій.

На сьогодні розрізняють три типи онтологій: предметно-орієнтовані (Domain-oriented), орієнтовані на прикладну задачу (Task-oriented) та загальні онтології (Top-level).

Онтологія ПО містить таксономію понять, додаткові відношення, екземпляри класів і різні види обмежень (аксіом). Аксіоми встановлюють семантичні обмеження для системи відношень.

Мета онтології задач – зробити знання доступними для повторного використання. Онтології задач визначають ступінь використання знань в процесі логічного виведення [7].

Загальна онтологія описує категорії – поняття верхнього рівня. Прикладами є фізичні, функціональні, поведінкові поняття і відношення, які відносяться до загальнонаукових понять та відношень.

Останнім часом прийнято будувати єдину онтологію, яка містить відразу три наведених типи онтологій. Ієрархічно це виглядає так: як правило загальна онтологія знаходиться на верхньому рівні ієрархії, а онтології ПО та задач до неї під’єднюються. Такий підхід дозволяє цілісно розглядати всі задачі в межах ПО [60].

Часто набір аксіом (обмежень), що становлять онтологію, має форму теорії логіки першого порядку, де терміни словника є іменами унарних і бінарних предикатів, званих відповідно концептами і відношеннями. У простому випадку онтологія описує тільки ієрархію концептів, зв’язаних відношеннями успадкування та агрегації. У складніших випадках в неї додаються відповідні аксіоми для вираження інших відношень між концептами і для того, щоб обмежити їх інтерпретацію. Враховуючи вищесказане, онтологія є базою знань, що описує факти, які передбачаються завжди істинними в рамках певної спільноти на основі загальноприйнятого значення використовуваного словника [8].

Отже, під формальною моделлю онтології *O* розуміють трійку такого вигляду:

$$O = \langle C, R, F \rangle, \tag{1.1}$$

де *C* – скінченна множина понять (концептів, термінів) предметної області, яку задає онтологія *O*; *R* – скінченна множина відношень між концептами (поняттями, термінами) заданої предметної області; *F* – скінченна множина функцій інтерпретації (аксіоматизація, обмеження), заданих на концептах чи відношеннях онтології *O* [9, 61-64].

Для побудови онтологій використовують чотири моделі подання знань: фрейми для подання понять, семантичні мережі для подання відношень, логіка предикатів другого порядку для подання аксіом та продукційні правила для побудови правил виведення [10]. Семантичну мережу фреймів (концептів) називають концептуальним графом (КГ) [11, 12].

У загальному вигляді структура онтології являє собою набір елементів чотирьох категорій:

- поняття;
- відношення;
- аксіоми;
- окремі екземпляри.

Поняття розглядаються як концептуалізації класу всіх представників якоїсь сутності чи явища (наприклад, БМП, знаки на карті). Класи (або поняття) є загальними категоріями, які можуть бути впорядковані ієрархічно. Кожен клас описує групу індивідуальних сутностей, які об’єднані на підставі наявності загальних властивостей.

Поняття можуть бути пов’язані різного роду відношеннями (наприклад, довжина, розташування), які пов’язують класи і описують їх. Найпоширенішим типом відношення, що використовується у всіх онтологіях, є відношення категоризації, тобто віднесення деякого об’єкта до певної категорії.

Поряд із зазначеними елементами онтології в неї також входять так звані «екземпляри». Екземпляри – це окремі представники класу сутностей або явищ, тобто конкретні елементи певної категорії (наприклад, екземпляром класу Знак є знак БТР).

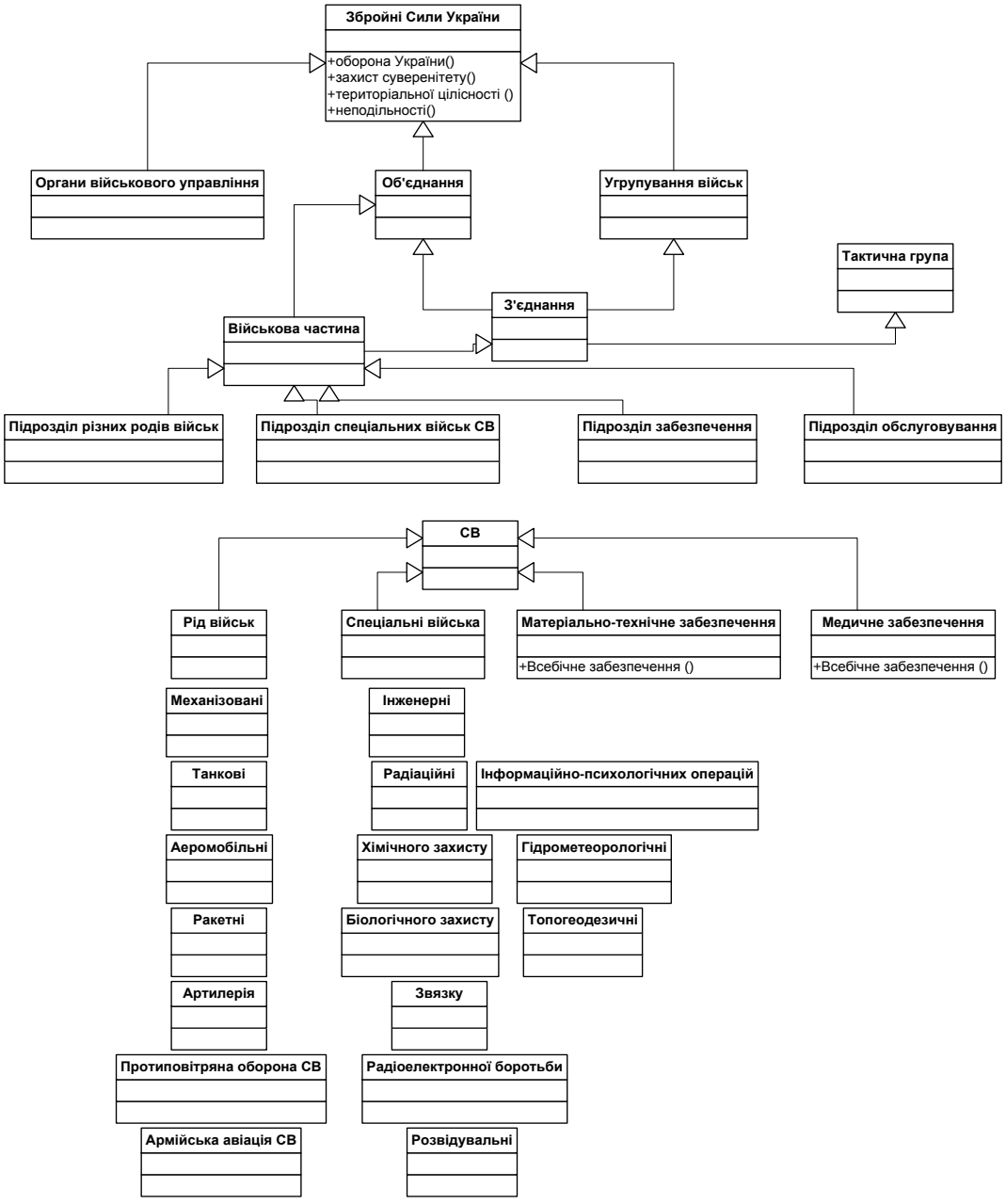

Рис. 1.3. Таксономія понять Збройних Сил України та сухопутних військ у вигляді діаграми класів UML

Складові онтології підпорядковуються своєрідною ієрархії. На нижньому рівні цієї ієрархії знаходяться екземпляри, конкретні індивіди, вище йдуть поняття, тобто категорії. На рівень вище розташовуються відношення між цими поняттями. Об’єднують ці всі елементи правила та аксіоми.

Наведемо приклад онтологічного підходу [65-70, 38-40, 46, 48]. Для побудови онтологічної моделі, насамперед, необхідно визначити ієрархію понять (множину *C*). Приклад такої таксономії понять, яка задає СВ ЗСУ, подану за допомогою діаграми класів UML, наведено на рис. 1.3. Інші діаграми наведені у 4-му розділі.

Функції інтерпретації *F* задають обмеження, або приймання певних значень властивостями концептів. Наприклад окремі концепти ВМР-1 приймають певні значення, зокрема максимальна швидкість 65 км/год.

Основним методом побудови інфологічної моделі у вигляді онтології є використання класифікацій. Класифікація – засіб впорядкування знань [71-78]. В об’єктно-орієнтованому аналізі визначення загальних властивостей об’єктів допомагає знайти загальні ключові абстракції й механізми, що, своєю чергою, приводить нас до простішої архітектури інфологічної моделі системи. На сьогодні не розроблені строгі методи класифікації й немає правил, що дають змогу виділяти класи й об’єкти. Немає таких понять, як «ідеальна структура класів», «правильний вибір об’єктів». Вибір класів є компромісним рішенням. Метою класифікації є визначення загальних властивостей об’єктів. Класифікуючи, ми поєднуємо в одну групу об’єкти, що мають однакову будову, або однакову поведінку.

Під час побудови онтології може виникнути ряд проблем. Під час класифікації існує можливість появи неоднозначностей в залежності від того, якій диференційній ознаці віддавати перевагу. Вибір того чи іншого рішення часом досить складно обґрунтувати. Найважливішим тут є вибрати який-небудь підхід і дотримуватися його протягом всієї роботи, тому варто задуматися над

низкою проблем і можливими універсальними методами їх усунення ще до початку прийняття рішень.

При створенні концептів виникає «проблема примітивності». Майже вся семантика та подання знань ґрунтуються на композиційній гіпотезі: можна визначити обмежений набір одиничних сутностей («атомів»), а всі інші («молекули») представляти як комбінацію (композицію) атомів. При цьому виникає питання: як багато таких «атомів» необхідно? У сучасних дослідженнях можна знайти два різних підходи: економний і неощадливий.

Економний підхід полягає у створенні малої кількості елементарних концептів, семантично простих, за допомогою яких можна пояснити значення більш складних понять. Тоді легко виявити зв’язність понять, однак складно утворювати нові сутності. Семантичні примітиви – це лексичні одиниці, що виражають елементарні, базові значення. Кількість таких одиниць не перевищує 100.

Неощадливий підхід дає змогу створювати будь-яку кількість індивідуальних сутностей онтології. Ця кількість може варіюватися від 10 до 100 000 і більше. Тут важко визначати зв’язність понять. Тобто такий підхід супроводжується, по суті, відмовою від композиційної гіпотези, але такий підхід має й перевагу, а саме легко формувати складні сутності.

Важко сказати, який підхід кращий, все залежить від того, як багато сутностей або наскільки складна предметна область. Підсумовуючи, слід сказати, що сучасна практика показує, що економного підходу дотримуються в основному формалісти, а користувачі схиляються до неекономного.

Побудова онтології складний процес, не завжди легко виділити поняття та диференційні ознаки. Існує кілька варіантів дій, що залежать від конкретних завдань і вихідного матеріалу. Існує можливість збору елементів для онтології безпосередньо. При такому підході спочатку збираються і класифікуються поняття, потім визначається відношення між ними. Універсальна вимога до онтології полягає в тому, що загальна структура онтології повинна бути

зрозумілою і повинна існувати можливість її багаторазового використання. Інші вимоги такі:

- зрозумілість: онтологія повинна бути зрозумілою й легко передавати зміст предметної області; вона повинна бути об’єктивною;
- послідовність: у ній повинні міститися твердження, які не суперечать одне одному, ієрархії понять (які зв’язані відношеннями), екземплярам.
- можливість розширення: наявність можливості введення нових елементів;
- мінімальна ступінь спеціалізації онтології: небажаність повного підпорядкування онтології конкретній задачі, що може ускладнити її подальше використання в інших задачах.

Не можна стверджувати, що цей список вимог до онтології є вичерпним, але він може допомогти при прийнятті тих чи інших рішень, що стосуються побудови інфологічної моделі ПО.

Існують формалізовані і докладні описи стандартів для онтологій. Наприклад Expert Advisory Group on Language Engineering Standards (http://www.ilc.cnr.it/EAGLES96/home.html), International Standard for Language Engineering (http://www.mpi.nl/ISLE/), The Language Technology Resource Center (http://flrc.mitre.org/References/Standards/). Використання стандартів має стати запорукою того, що створений інформаційний ресурс буде легко впроваджуватися у вже існуючі системи і враховуватиме всі особливості інформаційних технологій.

1.6. Огляд імітаційних моделей функціонування тактичних ланок

Для найбільш розвинених країн світу, таких, як США, Австралія, Великобританія, Франція, імітаційне моделювання бойових дій стало потужним інструментом, який надає можливість здійснювати оперативну підготовку органів управління в динамічних умовах, максимально наближених до бойових. Воно дозволяє виконувати тренування багато разів і при цьому уникнути застосування підпорядкованих сил і засобів, що у кінцевому підсумку веде до

скорочення витрат на проведення заходів оперативної підготовки та запобігання втрат особового складу та пошкодження озброєння та військової техніки.

Розвиток засобів імітаційного моделювання у США та їх застосування у військовій сфері розпочалися практично із початком розвитку засобів обчислювальної техніки. Вже у середині 1970-х років створена у складі міністерства оборони США Лабораторія імітації конфліктів розробила реалістичне програмне забезпечення для імітації бойових дій. Наприкінці 1970-х років була розроблена програма «Янус» – перша, в якій був використаний графічний інтерфейс. Вона застосовувалась у плануванні бойових дій у Панамі, на Середньому Сході, у Сомалі, Боснії та інших міжнародних конфліктних ситуаціях. У 1997 р. було створено найпотужнішу програму «Лівермо», що об'єднала та підсилила можливості двох попередніх програм – «Об'єднаної конфліктної моделі» (покращеної версії «Янус») та «Об'єднаного тактичного імітатора».

1999 р. – система JCATS (Joint Conflict and Tactical Simulation) використовувалася для репетицій майбутніх бойових операцій у конфлікті в Косові, а також морською піхотою та ВМС США для планування та участі у навчаннях на території Сан-Франциско. Під час навчань програма JCATS аналізувала дії фахівців та тестувала в реальному часі ефективність повітряної та артилерійської атак.

Сьогодні США проводять підготовку за двома з трьох основних програм Міністерства оборони з Підготовки і трансформації: Національної програми підготовки ведення спільних операцій та Програми розвитку наукових досягнень. Третьою основною програмою є Програма спільного оцінювання, що здійснюється під керівництвом Відділу з управління персоналом та готовності Міністерства оборони. Національна підготовка з ведення спільних операцій передбачає використання дійсних, віртуальних і вигаданих моделей і тренажерів в межах інтегрованої мережі, що складається з 30 постійних

тренувальних центрів для забезпечення виконання завдань із ведення спільних операцій в найбільш реальних умовах. Програма забезпечує посилену підготовку для ведення спільних операцій за допомогою використання широкого спектра дійсного, віртуального та вигаданого навчального середовища.

Окрім того, ця програма забезпечує підготовку та акредитовану підтримку в загальному 700 військовослужбовців щорічно у 20 штатах. Завданням цієї програми на тривалий термін є об'єднання родів військ і служб, посередників і багатонаціональних компонентів партнерів коаліції для проведення операцій [22, 23].

У 2007 р. Австралія підключилася до об'єднаного національного центру навчання США через спеціально створений спільний комплексний навчальний центр, розташований поблизу Сіднея. Сучасні засоби зв'язку забезпечують зв'язок оборонних сил Австралії з американськими силами та можливість проводити спільні навчання на базі моделювання. Таке моделювання можна корегувати 24 години на добу, сім днів на тиждень. Такий підхід збройних сил Австралії спрямований на підвищення боєготовності та взаємозв'язку між США та Австралією, а також сприяє інтеграції натурних, віртуальних і конструктивних технологій навчання.

Розвиток засобів імітаційного моделювання у ЗСУ.

Вітчизняне імітаційне моделювання як новий науковий напрямок, почало інтенсивно розвиватися наприкінці 60-х рр., коли стали широко впроваджуватися та використовуватися складні технічні системи у найрізноманітніших галузях людської діяльності (космос, бойові дії, транспорт, біологія, медицина, економіка, нові технології на виробництві й ін.).

Велися розробки й дослідження з різних напрямків в царині імітаційного моделювання:

- розвиток методології, методів і технологій моделювання;

- розробка засобів і систем моделювання на базі універсальних алгоритмічних мов моделювання;
- розробка пакетів моделювання широкого призначення;
- розробка проблемно-орієнтованих пакетів моделювання.

Імітаційне моделювання застосовувалося під час розв'язання завдань радіолокації, протиповітряної оборони, аналізу та розподілу ресурсів [133-136].

Закономірно, що розробники засобів і систем імітаційного моделювання в Україні використовували закордонний досвід розробки таких систем.

Історія становлення й розвитку імітаційного моделювання в Україні пов'язана з відповідними етапами у світовій практиці [55].

- етап 1 (1955-1960): програми для завдань моделювання розроблялися на основі таких загальновідомих універсальних мов, як FORTRAN та ALGOL;
- етап 2 (1961-1965): з'явилися перші мови моделювання: GPSS, SIMSCRIPT, SIMULA, CSL, SOL. Була розроблена так звана «концепція погляду на світ» (world view);
- етап 3 (1965-1970): з'явилося друге покоління мов моделювання GPSS V, SIMSCRIPT II.5, SIMULA 67;
- етап 4 (1971-1978): розвиток вже розроблених мов і засобів моделювання, орієнтований насамперед на підвищення ефективності процесів моделювання та перетворення моделювання у більш простий і швидкий метод дослідження складних систем;
- етап 5 (1979-1984): роки переходу від програмування до розвитку моделей, основний акцент було перенесено на ідентифікацію інтегрованих засобів імітаційного моделювання.

Процес моделювання включає такі етапи, як створення моделі, програмування, проведення імітаційних експериментів, обробку та інтерпретацію результатів моделювання. Однак традиційна перевага віддавалася етапу програмування. Така схема моделювання багато в чому

повторює схему проведення натурних випробувань і зводиться лише до імітації траєкторій вивчених моделей [32].

З появою імітаційних моделей змінилася концепція моделювання, яка тепер розглядається як єдиний процес побудови та дослідження моделей, що має програмну підтримку. Тепер головним стає формальне поняття моделі, яке не лише пояснює динаміку системи, але є предметом математичних досліджень. Стає можливим достовірний аналіз багатьох практично важливих властивостей моделі (стаціонарних розподілів, малих ймовірностей, чутливості, надійності й вірогідності результатів моделювання). Ці властивості особливо істотні при дослідженні високовідповідальних і великомасштабних систем, де ціна помилки особливо висока.

- етап 6 (1985-1994): перенесення програмного забезпечення для імітаційного моделювання на персональні ЕОМ з використанням засобів графічного інтерфейсу (для візуалізації та анімації процесів моделювання).
- етап 7 (1995-1998): розробка засобів технологічної підтримки процесів розподіленого імітаційного моделювання на мультипроцесорних ЕОМ і мережах.

Україна має більш ніж 25-річний досвід розробки та впровадження у різних ужиткових галузях засобів і систем імітаційного моделювання. Велика географія впроваджень свідчить про значний вплив зазначених розробок на рішення таких загальнодержавних і національних проблем, як:

- прийняття відповідальних проектних рішень у різних сферах людської діяльності;
- підготовка та навчання наукових і науково-технічних фахівців найбільш сучасним інструментам досліджень на базі ЕОМ;
- накопичення та використання досвіду досліджень у різних прикладних сферах у стандартній для всіх користувачів і дослідників формі.

На сьогоднішній день в Україні розпочато цикл робіт зі створення засобів і систем розподіленого імітаційного моделювання на платформі сучасних ЕОМ

та операційних систем. Досвід застосування імітаційного моделювання інших країн планується повною мірою застосовувати й в Україні.

Використання імітаційного моделювання значно розширює можливості ЗСУ щодо відпрацювання питань взаємодії з оборонними відомствами країн найближчого оточення та країн-партнерів. Створення центрів імітаційного моделювання у ЗСУ, що вже знаходяться в процесі своєї реалізації, дасть змогу системного проведення комп'ютерних навчань зі штабами бригад та інших військових формувань [23].

Одним із кроків у бік інтегрування імітаційного моделювання у процес оперативної і бойової підготовки ЗСУ є приєднання до одного з перспективних проектів в цій галузі – проекту SEESIM (Південно-східноєвропейського мережного моделювання). Проект SEESIM є однією з основних ініціатив, що реалізуються в рамках діяльності Ради міністрів оборони країн Південно-східної Європи (SEDM) за підтримки об'єднаного командування США в Європі. Навчальними цілями проекту SEESIM визначені:

- відпрацювання процедур взаємодії органів державного та військового управління під час виникнення надзвичайних ситуацій та загрозі скоєння терористичних актів;
- тренування персоналу органів державного та військового управління;
- відпрацювання процедур оповіщення під час виникнення надзвичайних і кризових ситуацій;
- перевірка функціонування всіх наявних засобів зв'язку, включаючи комерційні телефонні та факсимільні лінії зв'язку, Інтернет, PIMS тощо.

У рамках проекту один раз на два роки проводяться комп'ютерні командно-штабні навчання «SEESIM» на територіях семи країн-учасниць.

Іншим шляхом інтегрування імітаційного моделювання у процес оперативної і бойової підготовки ЗСУ є збільшення питомої ваги учасників з українського боку у щорічних навчаннях із використанням ресурсів імітаційного моделювання «Rapid Trident» (участь у міжнародній миротворчій

операції зі встановлення миру). Основною метою цього навчання є поліпшення сумісності між штабами та підрозділами країн-учасниць навчання, практичне використання досвіду, отриманого в миротворчих операціях та попередніх навчаннях, підвищення рівня взаєморозуміння та співробітництва між військовослужбовцями різних країн.

Істотним кроком впровадження імітаційного моделювання у процес бойової та оперативної підготовки стане розробка та проведення міжвідомчого, за участю представників МВС, МНС, ДПСУ, СБУ, командно-штабного навчання з використанням ресурсів імітаційного моделювання.

У відповідності до оборонних завдань, які покладені на ЗСУ щодо виконання завдань у районах надзвичайних ситуацій природного та техногенного характеру, в тому числі в умовах правового режиму надзвичайного стану, від ЗСУ можуть залучатись органи управління та військові частини Об’єднаних сил швидкого реагування. Участь у проведенні таких навчань допоможе не лише відпрацювати питання процедур взаємодії між відомствами, а й виявити в ході навчання такі проблемні питання, які не можливо передбачити в ході підготовки до виконання цих завдань.

На сьогоднішній день вже створений та функціонує центральний елемент системи імітаційного моделювання ЗСУ – Центр імітаційного моделювання кафедри інформатизації штабів Національної академії оборони України, де встановлена американська система JCATS.

Програма являє собою імітатор конфліктних ситуацій на землі, у повітрі й на морі, що можуть розв’язуватися на рівні як окремого солдата, так і корпусу швидкого реагування чисельністю до 5 тис. осіб. Комп’ютерна мережа розгорнута у 12 навчальних аудиторіях. Одночасно у цій мережі можуть працювати 179 військовослужбовців. Оперативна ситуація, що моделюється, є максимально наближеною до реальної: прораховуються сотні параметрів до найдрібніших деталей: місцевості, окремо схеми кожного будинку, стану

шляхів, погодних умов, тривалості світлового дня, використання різноманітної зброї і систем наведення [25].

Подальший розвиток передбачає створення подібних центрів у всіх видових інститутах та об'єднання їх у єдину мережу моделювання UASIMNET (Мережа імітаційного моделювання ЗСУ) (рис. 1.4). Це створить умови для проведення масштабних розподілених командно-штабних навчань із використанням ресурсів імітаційного моделювання, участь в яких одночасно можуть брати як штаб армійського корпусу, так й визначені штаби підпорядкованих бригад, що створює унікальні умови для відпрацювання питань злагодження штабів різних ланок [26].

Рис. 1.4. Проект мережі імітаційного моделювання ЗС України UASIMNET

Таким чином, зазначені шляхи інтегрування імітаційного моделювання в процес оперативної і бойової підготовки ЗСУ дозволять:

- досягнути якісно нового її рівня;
- залучити до оперативної підготовки більшу кількість з'єднань і частин;
- запобігти втрат у особовому складі, завдання шкоди навколишньому середовищу та значних витрат на підготовку і проведення навчань із органами управління з'єднань ЗСУ.

1.7. Висновки до розділу 1

У цьому розділі відображено такі результати:

- проаналізовано сучасні підходи до розроблення систем підтримки прийняття рішень у конкурентному середовищі, а саме у військовій сфері. Обгрунотовано використання у таких системах петлі OODA (Observation-Orientation-Decision-Action), запропоновану Джоном Бойдом. Така модель передбачає багаторазове повторення петлі, яка складаються з чотирьох послідовних взаємодіючих процесів: спостереження (observation); орієнтація (orientation); прийняття рішення (decision); дія (action);
- знання, які використовуються в цій предметній області є об’єктивними й містяться у нормативних документах (військовий статут, тактико-технічні показники, нормативні показники тощо). Тому запропоновано в якості ядра бази знань системи підтримки прийняття рішень використати онтологію. Актуальність створення онтології військових технологій також обумовлюється необхідністю підвищення ефективності вирішення інформаційно-аналітичних завдань і взаємодії фахівців у процесі обґрунтування та контролю реалізації заходів програми розвитку ЗСУ.

РОЗДІЛ 2. ОНТОЛОГІЧНИЙ ПІДХІД ДО ПОБУДОВИ СпППР КОМАНДИРІВ ТАКТИЧНИХ ЛАНОК НА ОСНОВІ ПЕТЛІ БОЙДА

У другому розділі розроблено метод моделювання петлі Бойда у військових застосування з використанням онтологічного підходу. Обґрунтовано взаємодію онтології із етапами петлі Бойда. В якості етапів петлі використано задачі розвідування території, імітаційного моделювання перебігу бою, визначення цілерозподілу наявних засоблів ураження за розвіданими цілями противника, корегування обстрілів противника. Під час пошуку ефективного цілерозподілу запропоновано використати генетичні алгоритми.

2.1. Поняття онтології військових технологій і особливості її побудови

Під онтологією військових технологій [88] в даному випадку розуміється формальна концептуалізація галузі військових технологій, однозначно сприйнята всім співтовариством практики (community of practice – *Cop*) і надається наступною моделлю

$$Cop \rightarrow O = <C, R, F>, \qquad (2.1)$$

де *Cop* – співтовариство практики (сукупність фахівців Міноборони, військово-промислового комплексу, науково-дослідних організацій, що беруть участь в обґрунтуванні та реалізації заходів програми розвитку БВТ), «→» – знак єдності сприйняття [19].

Область військових технологій характеризується відсутністю нормативно встановлених визначень і строгої класифікації технологій. Військові технології постійно розвиваються, що відбивається у розширенні та зміні понятійної системи.

Побудова формальної онтології, що включає аксіоматичну складову, для такої предметної області є надзвичайно складним завданням. Сфера військових технологій являє собою складно-структуровану область, що включає як абстрактні, узагальнюючі поняття, так і прикладну термінологію, яка містить поняття по конкретних реалізаціям військових технологій. У структурі онтологічної моделі військових технологій пропонується виділити чотири основних рівні.

Перший рівень – онтологія подання знань. Мета першого рівня – створення мови для специфікації онтологій більш низьких рівнів.

Оскільки область військових технологій є підкласом області технологій, то була введена відповідна онтологія верхнього рівня. Онтологія верхнього рівня може використовуватися як основа для побудови онтологій різних предметних областей. Вона описує основні поняття в галузі технологій, такі як «технологія», «знання», «технофакт», «технологія продукції», «виробнича технологія», «подвійна технологія», «виріб» та ін.

Третій рівень містить онтологію предметної області – військових технологій. До основних концептів області військових технології належать: «військова технологія», «технологія озброєння», «технологія виробництва озброєння», «базова військова технологія», «критична військова технологія», «перелік базових та критичних військових технологій», «програма розвитку базових військових технологій».

Прикладні онтології військових технологій, складові четвертого рівня, описують множину реалізацій військових технологій. Вони містять специфічну інформацію – концепти і відносини, що розкривають особливості певних видів зброї і військової техніки (лазерна зброя, динамічний захист, гіперзвуковий носій, комп’ютер на основі перепрограмованих логічних матриць, система навігації та ін.)

Розглянемо докладніше онтологію предметної області – онтологію військових технологій [49-52]. Військові технології являють собою технології,

призначені для використання у військовій сфері [20]. Військові технології поділяють на технології продукції – озброєння і виробничі технології – виробництва озброєння .

Сутність базових та критичних військових технологій пояснюється на основі найпростішої моделі збройної боротьби, що базується на положеннях теорії Джона Бойда [1]. Відповідно до цієї теорії передбачається наявність мінімум двох протиборчих сторін («Червоні» та «Сині»), які ведуть збройну боротьбу в певній сфері (поле бою). Формально під полем бою будемо розуміти багатовимірний простір, у межах якого воюючі сторони ведуть бойові дії [53, 54, 129].

Модель поля бою, безсумнівно, дуже важлива для опису технологічних особливостей сучасної війни і вибору технологій для ведення бойових дій. На війні завжди існувала точка, де стикалися війська. З появою артилерії, точка ця розрослася, і виникло поняття фронту. У Першу світову війну рили траншеї, що тяглися на сотні кілометрів. Потім з’явилася авіація і ракети, що ще більш збільшило ширину і глибину району бойових дій. В даний час все більша увага приділяється моделюванню війни в містах, в гористій і лісистій місцевості, обліку особливостей поля бою мережево-центричної війни. Сучасні військові фахівці говорять про «арени бойових дій», про «бойовий простір» і про те, що у війнах майбутнього поняття «поле бою» ще більш розшириться. Крім традиційних сфер битв (море, суша і повітряний простір) «бойовий простір» поступово захоплює космос, кіберпростір, соціальну та когнітивну сфери [21].

Обидві протиборчі сторони «Червоні» та «Сині» діють і приймають рішення в рамках циклу Бойда OODA – спостереження, орієнтація, рішення, дія.

У простому випадку, за інших рівних умов двох сторін, існують два основні способи досягнення перемоги. Перший шлях – зробити в кількісному вимірі свої цикли швидшими, що дозволить перехопити ініціативу, нав’язати противнику хід бою. Другий шлях – поліпшити якість прийнятих рішень, тобто

приймати рішення, більшою мірою відповідні ситуації, що складається, ніж рішення противника. Перемагає в даній моделі та сторона, у якої комбінація показників швидкості і якості військової діяльності вище.

Кожному елементу циклу OODA відповідає певна задача, рішення якої забезпечується складною сукупністю військових технологій – макротехнологій військової діяльності. Від ефективності макротехнологій залежить час і якість вирішення завдань на кожному етапі циклу і циклічної діяльності в цілому.

Військові технології реалізуються в конкретних зразках озброєння, складових системи озброєння. Тому рівень розвитку БВТ відіграє вирішальну роль у визначенні можливостей і якості системи озброєння протиборчих сторін.

Серед технологій, що входять до складу базових військових технологій, виділяють найбільш важливі, так звані, критичні військові технології, які забезпечують вирішення принципово нових військово-технічних завдань, істотний приріст ТТХ виробів військової техніки, або значне зниження витрат на їх експлуатацію.

2.2. Використання онтологій в петлі OODA

Розглянемо детальніше кожний етап петлі OODA у процесі його взаємодії з онтологією предметної області та задач, які в цій області виникають [33, 70] (рис. 2.1).

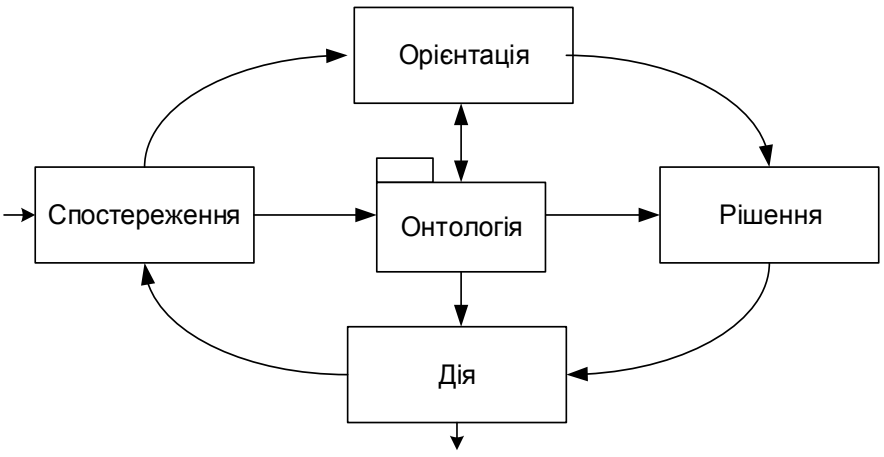

Рис. 2.1. Процеси петлі OODA при взаємодії з онтологією

2.2.1. Етап спостереження

Етап спостереження дає змогу здійснювати процес розбудови онтології, а також аналізувати її з метою вибору релевантної інформації, яка потрібна на наступних етапах петлі OODA [45].

Аналіз онтології здійснюється на основі отримання розвідувальних даних. Розвідники передають інформацію про наявні засоби противника, які відшукуються в онтології, опрацьовуються й надходять командиру ТЛ.

Задача обробки розвідувальних даних в загальному випадку виглядає так: група розвідників передає сукупність повідомлень (в загальному випадку різнобічних); необхідно визначити їх правдоподібність (задається ймовірністю) в залежності від джерела надходження інформації. Ймовірність правдивості джерела інформації задає командир ТЛ. Якщо повідомлення про один й той самий об’єкт x надходять з двох різних джерел інформації, то загальна їх правдоподібність обчислюється за формулою Шортліффа:

$$P(x) = P_1(x) + P_2(x) - P_1(x)P_2(x). \qquad (2.2)$$

Розвідка поля бою це найважливіший елемент, що забезпечує перевагу сил в бою. Переважно тактична розвідка спрямована на створення сприятливих умов для організованого і своєчасного вступу в бій і успішного його проведення. Тому необхідно розробити програмне забезпечення для швидкої та ефективної передачі розвідувальних даних у штаб, а також для узагальнення відомостей про бойовий склад, положення, стан угруповань військ наземного противника, характер його дій і намірів, сильних та слабких сторін, а також ступінь та характер інженерного обладнання. Для збору та опрацювання розвідувальних даних нами розроблено мобільний застосунок «Military intelligence» розд. 4.

2.2.2. Математичне забезпечення перебігу бою

Для того, щоб визначити елементи, які необхідно зберігати в онтології бази знань СпППР, проаналізуємо математичні моделі, які використовують для моделювання бойових дій, наведені у роботах [27-29, 58, 59, 106].

З формальної точки зору, будь-який бій – це реальний процес, що відбувається в часі і в просторі, характеризується наявністю двох ворогуючих сторін, складом і чисельністю, які змінюються під взаємним впливом. Кожна сторона прагне виконати поставлене перед нею завдання, що досягається найчастіше нанесенням протилежній стороні необхідного числа втрат при допустимому зменшенні своєї чисельності. Кожна сторона складається з деякого числа елементів, що є учасниками бою. Залежно від масштабу бою в якості елементів можуть вибиратися: окремі бійці, окремі зразки ОВТ в одному бою, або підрозділи і частини в іншому.

Кожний такий елемент характеризується деякою сукупністю змінних величин, що є функціями часу і визначають характер його дії і положення в просторі. Конкретне значення цих величин в деякий момент часу називається станом елемента. Зміна станів елементів бою в часі, що відбувається у відповідності з конкретними закономірностями перебігу бою, становить реальну сутність бою. Бій – процес кінцевий і характеризується своїм результатом. З формальної точки зору результат бою можна визначити як сукупність станів всіх елементів в деякий момент часу, після якого кожен з цих станів не змінюється.

Математична модель бою задається двома множинами $Q=\{q_1,q_2,...,q_n\}$ та $U=\{u_1,u_2,...,u_m\}$, що визначають якісний і кількісний склад воюючих сторін. Для кожного елемента $q_i \in Q$ існує багатовимірна випадкова функція $\zeta_i(t)=\zeta\left(\zeta_{i1}(t),\zeta_{i2}(t),...,\zeta_{ir(i)}(t)\right)$ для $T_0 \le t \le T_1$, де T_0 і T_1, відповідно, позначають моменти початку і кінця бою. Випадкові функції $\zeta_{i1}(t),\zeta_{i2}(t),...,\zeta_{ir(i)}(t)$ називаються параметрами елемента q_i, l – реалізація

випадкової функції $\zeta_i(t)$: $\zeta_i^l(t)=\zeta_i\left(\zeta_{i1}^l(t),\zeta_{i2}^l(t),...,\zeta_{ir(i)}^l(t)\right)$. Зріз випадкової функції $\zeta_i(t)$ в заданий момент часу $T_0 \le t_z \le T_1$ називається станом елемента q_i й позначається через $C_i(t_z)$. Вектор $\zeta_i^l(t_z)=\left(\zeta_{i1}^l(t_z),\zeta_{i2}^l(t_z),...,\zeta_{ir(i)}^l(t_z)\right)$ задає стан елемента q_i в момент часу t_z для *l*-ї реалізації й позначається як $C_i^l(t_z)$. Сукупність $\left\{C_i^l(T_0)\right\}$ для всіх $i=1,2,...,n$ задає початковий стан елементів *Q* для *l*-ї реалізації. Аналогічно описуються елементи $U_j\ (j=1,2,...,m)$.

Сукупність $\left\{D_j^l(T_0)\right\}$ для всіх $j=1,2,...,m$ називають початковим станом сторони U для *l*-ї реалізації, а сукупність $\left\{D_j^l(T_1)\right\}$ – результатом бою сторони *U* для *l*-ї реалізації. Множини елементів $\left\{C_i^l(T_1)\right\}$ і $\left\{D_j^l(T_1)\right\}$ разом називають об'єктивним результатом бою для *l*-ї реалізації, а $\left\{C_i^l(T_0)\right\}$ і $\left\{D_j^l(T_0)\right\}$ – початковим станом бою для *l*-ї реалізації.

Для прикладу розглянемо моделювання танкового бою в тактичному масштабі. Такий масштаб дає змогу до елементів бою віднести окремі бойові засоби: танк, самохідні установки, протитанкові засоби. Параметри елементів характеризують їх розташування на місцевості, їх переміщення, характер їх діяльності і результат цієї діяльності. Зміна цих параметрів у часі визначається випадковими функціями часу, тобто деякими випадковими процесами.

В якості параметрів для обраних елементів бою приймаються такі випадкові функції від дійсного аргументу часу t:

$\eta_1(t)$ – функція боєздатності;

$\eta_2(t)$ – функція місця розташування;

$\eta_3(t)$ – функція швидкості;

$\eta_4(t)$ – функція характеру дії;

$\eta_5(t)$ – функція кількості боєприпасів.

Детальніше математичні моделі перебігу бойових дій, які нами використані в модулі імітаційного моделювання, наведено у книзі «Математичні моделі бойових дій» за редакцією П. Н. Ткаченка [150].

Для визначення важливості цілей противника використано модель адаптивної онтології, яку розробив д.т.н, проф. В. В. Литвин. Важливість цілі визначається шкодою, яку завдаємо противнику, знищивши цю ціль. Для градації цілей проведено опитування експертів військової галузі для отримання оцінки важливості елементів онтології за 10-бальною шкалою (1 – важливість цілі кулемет, 10 – важливість цілі командний пункт бригади), тобто $W \in [1,10]$. Важливість елемента онтології, який задає ціль противника, визначається як середнє арифметичне експертних оцінок. Тоді ціль противника з максимальною вагою, як елемент онтології, визначаємо за формулою (3):

$$C_{Z^*} = \arg\max_{C_Z} \left(\sum_{\tilde{C}_i \to C_Z} W_{\tilde{C}_i} + W_{C_Z} \right). \qquad (2.3)$$

Модель повинна давати алгоритмічний спосіб отримання наближених реалізацій цих функцій, що дає змогу надалі отримати наближені характеристики цих функцій для практичного їх використання. Ці реалізації здійснюються в трьох основних моделях:

- переміщення елементів;
- виявлення елементів (цілей);
- стрільби.

Для реалізації моделі побудуємо онтологію предметної області.

Термінами предметної області в даному випадку будуть: бойові машини, гармати, артилерійські снаряди тощо. Зв’язками між термінами будуть: «має снаряд», «має гармату» тощо.

Мета використання процедурних складових онтологічної моделі повинна полягати в тому, що на інформаційному рівні виконання операцій ідентифікації поточного стану об’єкту задаються «межі» проблемних ситуацій. На їх основі

формується поточний інформаційний образ процесу діяльності, який будемо називати апостеріорною моделлю.

2.2.3. Моделювання основних процесів бойових дій

Моделювання пересування. У моделі переміщення (пересування) реалізуються функція місця розташування і функція швидкості. Відмінною рисою всіх сухопутних боїв є те, що всякий такий бій відбувається на деякій місцевості, яка істотно впливає на його перебіг. У будь-якій стохастичній моделі місцевість можна враховувати двояко: 1) інформація про характеристики місцевості не випадкова і є частиною вихідної інформації для моделювання перебігу бою; 2) інформація про характеристики місцевості є випадковою, і конкретні значення цих характеристик в моделі враховуються методом статистичних випробувань. Вибір підходу під час моделювання залежить від мети дослідження та наявних даних для такого дослідження.

Отримати постійну інформацію про будь-яку реальну місцевість неважко, зокрема вона може бути безпосередньо взята з карти. Однак висновки, отримані на моделі з використанням такого підходу, можна розповсюдити на досить вузький клас різних типів місцевості. Другий підхід значно розширює цей клас, але отримати випадкові закони зміни характеристик місцевостей часто буває складно. Безперервне відображення місцевості в стохастичній моделі неможливе, бо навіть на картах інформація дається не для кожної точки, а узагальнено. Це відноситься до будь-якої інформації, окрім координат. Тому ділянку місцевості, на якій відбувається реальний бій, розіб’ємо на елементарні ділянки, кожна точка якої характерна тим, що вона має однакову властивість з іншими точками цієї ж ділянки.

Прийнятий наступний принцип розбиття місцевості на елементарні ділянки. Ділянка місцевості, на якій відбувається бій, розбивається на однакові за величиною квадрати зі стороною a_0. Вважається, що всі точки одного квадрата мають інформацію, однакову з центром квадрата.

Сукупність таких елементарних ділянок впорядкована, тобто кожній ділянці відповідає індекс (i, j), де i – номер вертикальної смуги, а j – горизонтальної. Між системою таких індексів та географічними координатами встановлено взаємно-однозначну відповідність, що дає змогу за індексом знаходити на карті відповідну ділянку. Для кожної ділянки з індексом (i, j) задається необхідна для моделі інформація, яка характеризує цю ділянку як елемент місцевості: тип рельєфу, характер природних і штучних споруд, прохідність. Кількість цих ознак залежить від виду та характеру задачі. Виходячи з цього, необхідно задати сукупність функцій від аргументу (i, j), множини значень яких будуть визначати кількісне значення ознак місцевості на кожній ділянці. Таким чином, інформація про місцевість на кожній ділянці визначається значеннями деякого числа ознак $f_1(i, j), f_2(i, j),\ldots, f_k(i, j)$.

При виборі конкретного значення швидкості переміщення бойової машини враховується його тип, характер ґрунту тих ділянок, по яких відбувається рух, і кут нахилу руху. Відомо, що максимальна швидкість руху бойової машини визначається за формулою:

$$V_{\max} = \frac{270 N_D \eta_T}{G\left(g\cos\alpha + \sin\alpha\right)},$$

де N_D – потужність двигуна бойової машини (БМ); $\eta_т$ – коефіцієнт корисної дії двигуна БМ, який враховує втрати потужності в трансмісії і ходовій частині; G – маса БМ, g – коефіцієнт супротиву ґрунту, α – кут нахилу ґрунту, $V_{\max}$ – максимальна швидкість. БМ

Значення g для кожної ділянки задано у вхідній інформації. При переміщенні з ділянки до ділянки береться півсума відповідних значень коефіцієнта g.

Моделювання виявлення цілей. Як правило, під час моделювання боїв підрозділів сухопутних військ вважають, що кожний елемент у процесі бою веде спостереження за елементами противника, які є цілями по відношенню до

елементів, що ведуть спостереження. Всі цілі залежно від відстані, різниці висот, рельєфу і рослинності умовно поділяються на дві групи: невидимі і видимі. Цілі першої групи виявити неможливо, цілі другої групи рано чи пізно виявляються. Це означає, що пошук цілі – випадкова подія і кожна ціль другої групи має певну ймовірність виявлення. Природно, що чим кращі умови спостереження і чим довше воно ведеться, тим більша ймовірність виявлення.

Конкретний вид функції ймовірності виявлення може бути самим різним – він залежить від типу модельованого реального бойового процесу. Приймемо, що ймовірність виявлення описується формулою: $p(t) = 1 - e^{-yt}$, де y – миттєва щільність ймовірності виявлення; t – час спостереження, $p(t)$ – ймовірність виявлення цілі.

Це означає, що ydt ймовірність виявлення цілі за досить малий відрізок часу dt. Функція y залежить від відстані до цілі, її розмірів, типів засобів спостереження, метеорологічних умов спостереження.

Однією із можливих реалізацій функції y є [27-28]:

$$y = k\frac{S}{r^2},$$

де S – площа проекції цілі на площину, перепендикулярну лінії спостереження; r^2 – відстань до цілі; k – коефіцієнт, який враховує всі інші фактори, що впливають на виявлення цілей.

Таким чином, кінцева формула для визначення ймовірності виявлення цілі буде такою:

$$p(t) = 1 - e^{-k\frac{S}{r^2}t}.$$

Для цілей другої групи за формулою ймовірності виявлення обчислюємо значення цих ймовірностей, а потім методом статистичних випробувань за цими значеннями вибираємо всі видимі цілі. Інформація про всі виявлені цілі використовується далі для моделювання стрільби.

Моделювання стрільби. У моделі стрільби шукається поточна інформація про результат стрільби по виявлених цілях, при цьому визначаються ймовірності ураження кожної цілі. Ймовірність ураження є функцією таких аргументів: відстані до цілі, її розмірів, швидкості руху засобу ураження і цілі.

Після ураження цілі засіб ураження переходить в стан спостерігача, завдяки цьому будується реалізація функції характеру дій. Розсіювання при стрільбі здійснюється за нормальним законом, де математичне сподівання – це координати цілі, а дисперсія – розсіювання по *x* і *y*. Знаючи координати попадання снаряда, радіус його ураження можна визначити ймовірність ураження цілі.

Якщо ціль уражена, то вона втрачає свою боєздатність.

Детальніше математичні моделі, які покладено в основі модуля імітаційного моделювання перебігу бою, наведено у [64, 77, 138, 141, 150].

2.2.4. Задача цілерозподілу по груповій цілі

Цілерозподіл – це операція, яка полягає в призначенні певної цілі певному вогневому засобу. Якщо в наявності декілька цілей, котрі потрібно піддати вогневій дії, а в нашому розпорядженні є декілька вогневих одиниць (літаків, гармат, ракет), то, вирішуючи завдання цілерозподілу, ми повинні точно вказати, які засоби, в якій кількості і коли направляються на кожну із цілей, яку належить обстріляти [27, 98, 99].

Рішення по цілерозподілу являє собою типовий приклад тактичного рішення.

В умовах минулих воєн, коли бойові дії не були такими швидкоплинними, рішення по цілерозподілу зазвичай приймалося командиром: на основі бойового досвіду і здорового глузду. В теперішній час такий спосіб не завжди може нас задовольнити. Нерідко зустрічаються умови бою, коли на це просто не вистачає часу (на приклад, при відбитті повітряної атаки). В таких умовах вирішення завдання перерозподілу приходиться передавати

автоматичному пристрою – обчислювальній машині. В інших випадках бойова обстановка настільки складна, кількість можливих варіантів настільки велика, що прийняття рішення без спеціальних розрахунків виявляється не під силу навіть досвідченому командиру. Щоб завдання цілерозподілу можна було передати машині, повинен бути створений алгоритм вирішення цього завдання (алгоритм цілерозподілу).

Розрізняють два варіанти задачі цілерозподілу:

- для засобів оборони;
- для засобів нападу.

Різниця їх у тому, що цілерозподіл засобів оборони здійснюється в ході самої операції (наприклад, при відбитті повітряного нальоту), умови якої наперед невідомі і залежать від противника. Цілерозподіл засобів нападу зазвичай проводиться завчасно при плануванні нальоту чи обстрілу. Якщо мова йде про неповністю розвіданих, або часто змінюючих дислокацію цілях, чітка різниця між тим чи іншим завданням зникає.

Завдання цілерозподілу в повному своєму об’ємі дуже складне і потребує врахування багатьох факторів, таких, наприклад, як:

- дислокація засобів розподілу, їх бойова готовність;
- пропускна здатність каналів наведення і каналів зв’язку;
- глибина зони, що проглядається радіолокаційними системами;
- можливість застосування противником маневру, перешкод і т. д.

Всі ці обставини так чи інакше враховуються при складанні алгоритму цілерозподілу, який попередньо тестується на моделях бою, щоб вибрати найбільш кращий варіант. Як завжди при вирішенні складних завдань дослідження операцій, тут не вдається обмежитися оцінкою по одному-єдиному критерію (показнику) ефективності, а шукається компромісне вирішення, що задовольняє цілий ряд критеріїв.

Під час цілерозподілу потрібно так розподілити засоби ураження за цілями, щоб деякий критерій досягав максимуму. Як це зробити? Самий

елементарний спосіб – це спосіб повного перебору: перебираються всі можливі варіанти розподілу засобів за цілями і вибирається той із них, при якому критерій оптимальності досягає максимуму. У випадку, коли цілей і засобів багато, простий перебір всіх можливих варіантів стає складним, навіть при машинній реалізації. Виникає завдання розв'язування задачі цілерозподілу, не перебираючи всіх можливих варіантів. Запропоновано багато способів вирішення задачі цілерозподілу, які скорочують кількість перебирань (зокрема метод лінійного програмування). Існують й інші способи. Оригінальним способом вирішення завдання цілерозподілу є метод Монте-Карло, коли засоби ураження випадковим чином розподіляються по цілях, і з усіх розподілень вибирається найвигідніший [98].

Нами пропонується для розв'язування задачі цілерозподілу використати методи штучного інтелекту, а саме генетичні алгоритми [99]. Такі алгоритми використовуються для вирішення задач оптимізації і моделювання шляхом послідовного відбору, комбінування і варіації шуканих параметрів з використанням механізмів, які схожі на біологічну еволюцію.

Постановка задачі. Розроблення методу ефективного (близького до оптимального) цілерозподілу на основі використання генетичних алгоритмів.

Критерії цілерозподілу. Нехай в нашому розпорядженні є n засобів і нам потрібно обстріляти роззосереджену групу, що складається з N цілей. Кожен засіб здійснює лише один постріл і може в принципі стріляти по кожній цілі, але не з однаковою ефективністю.

Ймовірність ураження i-м засобом j-ї цілі задана і рівна P_{ij}. Значення P_{ij} записані у табл. 2.1.

Для визначення цих ймовірностей використовуються таблиці з нормативних документів, які зберігаються в таблицях БД. З якої таблиці БД використовувати дані визначається онтологією СВ ЗСУ на основі опрацювання параметрів (видимість, прохідність, швидкість, боєздатність).

Таблиця 2.1. Розподіл засобів та цілей

Номер засобу i	Номер цілі j			
	1	2		N
1	p_{11}	p_{12}	. . .	p_{jN}
2	p_{21}	p_{22}	. . .	p_{2N}
. . .	. . .	. . .	. . .	. . .
n	p_{n1}	p_{n2}		p_{nN}

Необхідно знайти оптимальний (найкращий) цілерозподіл, призначивши кожному засобу певну ціль, по якій вона повинна стріляти (при цьому можливо, що одна і та ж ціль буде обстріляна кількома засобами). Назвемо поставлене завдання завданням цілерозподілу $n \times N$.

Щоб вирішити завдання цілерозподілу, необхідно перш за все вибрати показник ефективності. Таким показником в залежності від умов стрільби може бути:

1) математичне сподівання числа уражених цілей (ймовірність того, що у складі групи буде уражено не менше заданого числа цілей);
2) ймовірність того, що будуть уражені всі без винятку цілі, і т. п.

У першому випадку необхідно намагатись знищити в середньому якнайбільше цілей, і показником ефективності буде математичне сподівання кількості уражених цілей:

У другому випадку в якості показника ефективності вибирається ймовірність ураження всіх без винятку цілей P_N.

У відповідності з цими двома основними критеріями розрізняють:

1) цілерозподіл «за математичним сподіванням»;
2) цілерозподіл «за ймовірністю».

Показником ефективності цілерозподілу за математичним сподіванням є величина:

$$M_n = M[X_n], \tag{2.4}$$

де випадкова величина X_n – кількість уражених цілей.

Під час стрільби по груповій цілі середнє число уражених цілей дорівнює сумі ймовірностей ураження окремих елементарних цілей (одиниць):

$$M_n = p_1 + p_2 + ... + p_N, \tag{2.5}$$

де p_1 – ймовірність ураження першої цілі; p_2 – ймовірність ураження другої цілі; p_N – ймовірність ураження N-ї цілі. Тим самим ми отримуємо задачу:

$$M_n = \sum_{j=1}^{N} p_j \to \max. \tag{2.6}$$

Отже, під час цілерозподілу за математичним сподіванням потрібно так розподілити засоби ураження за цілями, щоб сума ймовірностей ураження досягала максимуму.

На практиці може виявитися, що деякі окремі цілі (наприклад, установники перешкод чи носії ядерних засобів, якщо їх вдається виявити) важливіші для нас, ніж останні. Якщо цілі нерівноцінні, то їм можуть бути приписані різноманітні «ваги» – k_j, а показником ефективності буде не просто середнє число уражених цілей, а «зважене» середнє:

$$M_n = \sum_{j=1}^{N} k_j \cdot W_j \to \max.$$

Розглянемо другий принцип цілерозподілу «за ймовірністю», коли показником ефективності є ймовірність уразити всі без винятку цілі P_N. Очевидно, що коли кількість засобів ураження менша кількості цілей N, то $P_N = 0$.

Щоб P_N не було рівним нулю, очевидно, необхідно мати можливість обстріляти всі цілі, тобто повинна виконуватися умова $n>N$.

Ймовірність ураження усіх цілей виражається добутком ймовірностей ураження окремих цілей:

$$P_N = p_1 \cdot p_2 \cdot \ldots \cdot p_N . \qquad (2.7)$$

Необхідно так розподілити засоби ураження по цілях, щоб ймовірність P_N була максимальною:

$$P_N \to \max . \qquad (2.8)$$

Можливі такі випадки:

1. Кожна із цілей уражується кожним із засобів приблизно з однаковою ймовірністю p. При такій умові завдання цілерозподілу спрощується. Питання стоїть не про те, який засіб ураження направити по якій цілі, а скільки засобів і по якій із цілей? У роботі [99] доведено, що максимальне математичне очікування кількості уражених цілей M_n так само, як і максимальна ймовірність ураження всіх цілей P_N, досягається у тому випадку, якщо засоби ураження розподілити між цілями найбільш рівномірно. Наприклад, якщо десять засобів обстрілюють п’ять цілей, то потрібно на кожну ціль виділити по два засоби ураження; якщо дванадцять засобів обстрілюють п’ять цілей, то необхідно на кожну ціль виділити два засоби ураження, а ті дві, що залишилися призначити будь-яким чином на будь-які дві цілі.

2. Якщо стоїть умова обов’язкового обстрілу всіх цілей, то завдання цілерозподілу спрощується за рахунок скорочення кількості варіантів. А сама задача цілерозподілу зводиться до задачі про призначення.

Для цих двох випадків генетичних алгоритмів не потрібно. Для всіх інших випадків скористаємось цими алгоритмами.

Генетичні алгоритми використовуються для розв’язування оптимізаційних задач на основі механізмів селекції та репродукції. Коротко

пояснимо принципи генетичних алгоритмів (детальніше можна прочитати у роботах [101-103]). Механізм селекції полягає у виборі хромосом (у нашому випадку це вектор довжини n, номер елемента якого задає наш засіб знищення (засіб ураження), а значення елементу – ціль противника) з найвищою оцінкою (тобто найбільш пристосованих), які репродукують частіше, ніж особини з більш низькою оцінкою (гірше пристосовані). Репродукція означає створення нових хромосом у результаті рекомбінації генів (елементів вектора) батьківських хромосом. Рекомбінація – це процес, в результаті якого виникають нові комбінації генів. Для цього використовуються дві операції: схрещування, що дозволяє створити дві зовсім нові хромосоми нащадків шляхом комбінування генетичного матеріалу пари батьків, а також мутація, яка може викликати зміни в окремих хромосомах.

Генетичний алгоритм складається з наступних кроків:

1) ініціалізація, або вибір вихідної популяції хромосом;
2) оцінка пристосованості хромосом в популяції;
3) перевірка умови зупинки алгоритму;
4) селекція хромосом;
5) застосування генетичних операторів;
6) формування нової популяції;
7) вибір «найкращої» хромосоми.

На кожній ітерації генетичного алгоритму пристосованість кожної особини даної популяції оцінюється за допомогою функції пристосованості, і на цій основі генерується наступна популяція особин, що складає множину потенційних рішень оптимізаці заційної задачі. Чергова популяція в генетичному алгоритмі називається поколінням, а до новостворюваної популяції особин застосовується термін «нове покоління», або «покоління нащадків». В якості функції пристосованості (у літературі з генетичних алгоритмів її ще називають fitness function) у нашому випадку виступає

формула (2.6). Якщо кілька засобів ураження обстрілюють одну ціль, то ймовірність знищення такої цілі зростає згідно формули:

$$p_i = 1 - \prod_{j \in Z_i} (1 - p_j), \qquad (2.9)$$

де Z_i – множина засобів ураження, які обстрілюють *i*-ту ціль.

Для реалізації запропонованого підходу обрано БД реляційного типу (а саме MySQL), в якій зберігається інформація про наші наявні засоби ураження, розвідані цілі противника та матриця ймовірностей знищення цілей певним вогневим засобом. Хромосома являє вектор, де номер елемента вектора – відповідає ключу нашому засобу знищення в БД, а значення елемента – ключ цілі в БД.

Ініціалізація, тобто формування вихідної популяції, полягає у випадковому виборі заданої кількості хромосом (ми брали 50) розмірності *n*, що представляють випадковий розподіл цілей за засобами ураження.

Оцінювання пристосованості хромосом в популяції полягає в розрахунку функції пристосованості для кожної хромосоми цієї популяції. Чим більше значення цієї функції, тим вища «якість» хромосоми.

Визначення умови зупинки генетичного алгоритму залежить від значення функції пристосованості (наприклад досягнення очікуваного оптимального значення). Алгоритм також зупиняється, коли його виконання не приводить до покращення вже досягнутого значення, або після виконання заданої кількості ітерацій. Якщо умова зупинки виконана, то проводиться перехід до завершального етапу вибору «найкращої» хромосоми. В іншому випадку на наступному кроці виконується селекція.

Селекція хромосом полягає у виборі (за розрахованими на другому етапі значеннями функції пристосованості) тих хромосом, які братимуть участь у генерації нащадків для наступної популяції, тобто для чергового покоління. Такий вибір здійснюється згідно з принципом природного відбору, за яким

найбільші шанси на участь у генерації нових особин мають хромосоми з найбільшими значеннями функції пристосованості.

У результаті процесу селекції генерується батьківська популяція (matingpool). Застосування генетичних операторів до хромосом, відібраних за допомогою селекції, призводить до формування нової популяції нащадків від генерованої на попередньому кроці батьківської популяції.

У генетичному алгоритмі застосовуються два генетичних оператора: оператор схрещування (crossover) та оператор мутації (mutation). Схрещування у генетичному алгоритмі здійснюється практично завжди, тоді як мутація – досить рідко. Ймовірність схрещування ми задали 0.8, а мутації – 0.2.

На першому етапі схрещування вибираються пари хромосом з батьківського популяції. Це тимчасова популяція, що складається з хромосом, відібраних в результаті селекції та призначених для подальших перетворень операторами схрещування і мутації з метою формування нової популяції нащадків. На даному етапі хромосоми з батьківського популяції об'єднуються в пари. Це здійснюється випадковим способом. Далі для кожної пари відібраних таким чином батьків розігрується позиція гена (локус) у хромосомі, що визначає так звану точку схрещування. Якщо хромосома кожного з батьків складається з n генів, то очевидно, що точка схрещування n_k представляє собою натуральне число, менше n. Тому фіксація точки схрещування зводиться до випадкового вибору числа з інтервалу $[1, n-1]$. У результаті схрещування пари батьківських хромосом виходить така пара нащадків: нащадок, хромосома якого на позиціях від 1 до n_k складається з генів першого з батьків, а на позиціях від n_k+1 до n – із генів другого з батьків.

Оператор мутації змінює значення гена в хромосомі. У нашому випадку випадково обирається елемент вектора, який випадково змінюється на деякий ключ цілі з БД.

Після операторів схрещування й мутації генерується нова популяція. Хромосоми, отримані в результаті застосування генетичних операторів

додаються до складу нової популяції. Вона стає так званою поточною популяцією для даної ітерації генетичного алгоритму.

На кожній черговій ітерації розраховуються значення функції пристосованості для всіх хромосом цієї популяції, після чого перевіряється умова зупинки алгоритму і (або) фіксується результат у вигляді хромосоми з найбільшим значенням функції пристосованості, або здійснюється перехід до наступного кроку генетичного алгоритму, тобто до селекції. Вся попередня популяція хромосом замінюється новою популяцією нащадків, яка має ту ж чисельність.

Кращим розв’язком вважається хромосома з найбільшим значенням функції пристосованості.

Для моделювання обиралась певна кількість генерації поколінь хромосом. Результати експериментів показали, що при генерації 30 поколінь знайдена найкраща хромосома близька до оптимального цілерозподілу. Розроблений модуль цілерозподілу на основі генетичних алгоритмів входить в склад автоматизованої системи управління сухопутними військами [104].

Основною перевагою запропонованого підходу є значне зменшення складності алгоритму цілерозподілу. Складність повного перебору – експотенційна, а генетичного алгоритму – лінійна. Тим самим набагато прискорюється процес прийняття рішень. Особливо це важливо, коли події щодо цілерозподілу відбуваються в реальному часі. Хоча отриманий розв’язок цілерозподілу не є завжди оптимальним, однак він близький до оптимального, а виграш в часі отримання розв’язку є значним.

2.2.5. Методи математичного опису динаміки бойових дій

Розглядаючи динаміку бою численних угруповань, не можна строго зафіксувати моменти кожного пострілу кожної з бойових одиниць, що беруть участь в бою: модель виявилася б занадто складною, а головне не відображала б реального стану речей. Насправді моменти пострілів окремих бойових одиниць неминуче будуть випадковими.

Тому при розгляді динаміки бою ми всюди будемо користуватися схемою випадкового розподілу вогню. Послідовність пострілів у часі будемо розглядати як деякий потік подій (рис. 2.2).

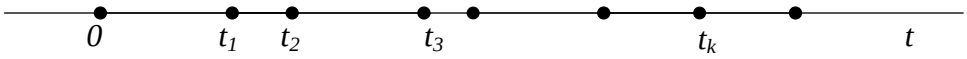

Рис. 2.2. Потік подій

У теорії ймовірностей потоком подій називається послідовність однорідних подій, що здійснюються одне за іншим в якості випадкових моментів часу (рис. 2.3.). Прикладами можуть служити:

- потік включень освітлювальних приладів в електромережу;
- потік відмов (збоїв) обчислювальної машини;
- потік повітряних цілей, що входять в зону дії системи ППО;
- потік атак, яким піддається повітряна ціль над оборонявся територією.

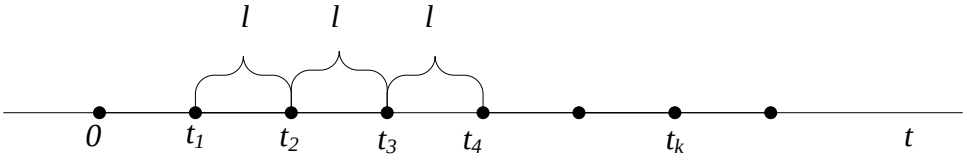

Рис. 2.3. Регулярний потік подій

З першого погляду може здаватися, що регулярний потік являє собою найбільш простий з усіх можливих потоків подій. Однак це не так, адже в регулярному потоці моменти появ послідовних подій пов'язані між собою жорсткою, функціональною залежністю. Це сильно ускладнює аналіз явищ, в яких фігурує одночасно велика кількість таких потоків (наприклад, бій численних груп бойових одиниць, кожна з яких здійснює постріли в моменти часу, розділені жорсткими інтервалами). Набагато простіше аналізувати

потоки подій, що з'являються у випадкові моменти часу незалежно один від одного (так звані пуассонівські потоки) [27].

Нехай, події A_1, A_2, …, з'являються в моменти t_1, t_2, …, які ми зобразимо точками на осі абсцис $0t$ (рис. 2.4). Припустимо, що точки з'являються на осі $0t$ поодинці, тобто збіг в один і той же момент часу двох або більше подій практично неможливий. Такий потік називається ординарним.

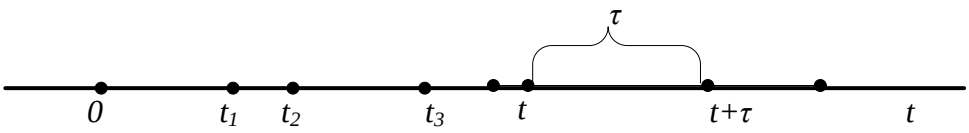

Рис.2.4. Потік подій

Виділимо на осі $0t$ відрізок довжиною τ, що починається в точці t та закінчується в точці $t+\tau$. Потік подій називається потоком без наслідків, якщо ймовірність того, що на ділянку τ попаде та чи інша кількість точок, не залежить від того, як розташовані інші точки на осі $0t$, що не попали на цю ділянку.

У теорії ймовірностей доводиться, що якщо потік подій ординарний і не має післядії, то число подій, що потрапляють на задану ділянку τ, розподіляється по так званому закону Пуассона. А саме, ймовірність того, що на ділянку потрапить рівно τ точок, визначається формулою:

$$P_m = \frac{a_m}{m!} e^{-a},$$

де a – середня кількість точок, що потрапляють на ділянку τ.

Щільність потоку подій може бути як постійною, так і змінної, що залежить від часу. Розглянемо спочатку найпростіший випадок, коли щільність потоку постійна: $\lambda = const$.

Такий потік подій називається стаціонарним пуассонівським, або найпростішим потоком. Очевидно, для найпростішого потоку середнє число подій, що припадає на ділянку τ, пропорційно його довжині: $a = \lambda\tau$ і не залежить від того, де саме на осі 0τ обрано ділянку.

У випадку, коли щільність потоку залежить від часу, тобто задана деякою функцією $\lambda(t)$, середнє число подій, що припадає на ділянку τ, виражається інтегралом: $a = \int_{t}^{t+\tau} \lambda(t)dt.$

Знаходження ймовірності того, що за час τ не з'явиться жодна подія виконується за формулою: $P_0 = e^{-a}$.

Ймовірність того, що на ділянці τ з'явиться хоча б одна подія: $R_1 = 1 - e^{-a}$.

Детальніше математичне забезпечення перебігу бою наведено у роботі [151].

2.3. Подання петлі Бойда у вигляді автомату Мура

Підсумовуючи вище наведене, розробимо модель петлі Бойда з використанням онтологій у вигляді автомату Мура.

Скінченний автомат можна подати в вигляді математичної схеми (F-схеми), яка характеризується шістьма елементами:

- скінченною множиною X вхідних сигналів (вхідний алфавіт);
- скінченною множиною Y вихідних сигналів (вихідний алфавіт);
- скінченною множиною Z внутрішніх станів (внутрішній алфавіт або алфавіт станів);
- початковим станом z_0, $z_0 \in Z$;
- функцією переходів $\varphi(z, x)$;
- функцією вихідів $\psi(z, x)$;

Автомат задається F-схемою $F = <Z, X, Y, \varphi, \psi, z_0>$. Він функціонує у дискретному автоматному часі, моментами якого є такти, кожному з яких

відповідають однакові значення вхідних і вихідних сигналів та внутрішнього стану. Автомат Мура – це *F* – автомат другого роду, для якого:

$$z(t+1) = \varphi[z(t), x(t)],\ t = 0, 1, 2,\ldots ; \qquad (2.10)$$

$$y(t) = \psi[z(t)],\ t = 0, 1, 2,\ldots ; \qquad (2.11)$$

тобто функція виходів не залежить від вхідної змінної $x(t)$.

Отриманий нами автомат Мура наведено на рис. 2.5. На цьому рисунку використано такі позначення:

$S = \{s_1,\ldots,s_5\}$ – множина станів автомата. $s_0 \in S$ – початковий стан, s_0 – етап петлі Бойда «Спостереження», s_1 – етап петлі Бойда «Орієнтація», s_2 – процес «Редагування онтології», s_3 – процесс «Пошук релевантних знань», s_4 – етап петлі Бойда «Рішення», s_5 – етап петлі Бойда «Дія».

x_1 – дані, яких немає в онтології, x_2 – дані про противника, x_3 – онтологічні дані (x_3' – для рішення, x_3'' – для дії), x_4 – оцінка обстановки, x_5 – моделювання ситуації, x_6 – синтез даних, x_7 – аналіз даних, x_8 – оцінка рішення, x_9 – збір даних, x_{10} – пропоноване рішення, x_{11} – редагування онтології (нові дані), x_{12} – навколишнє середовище.

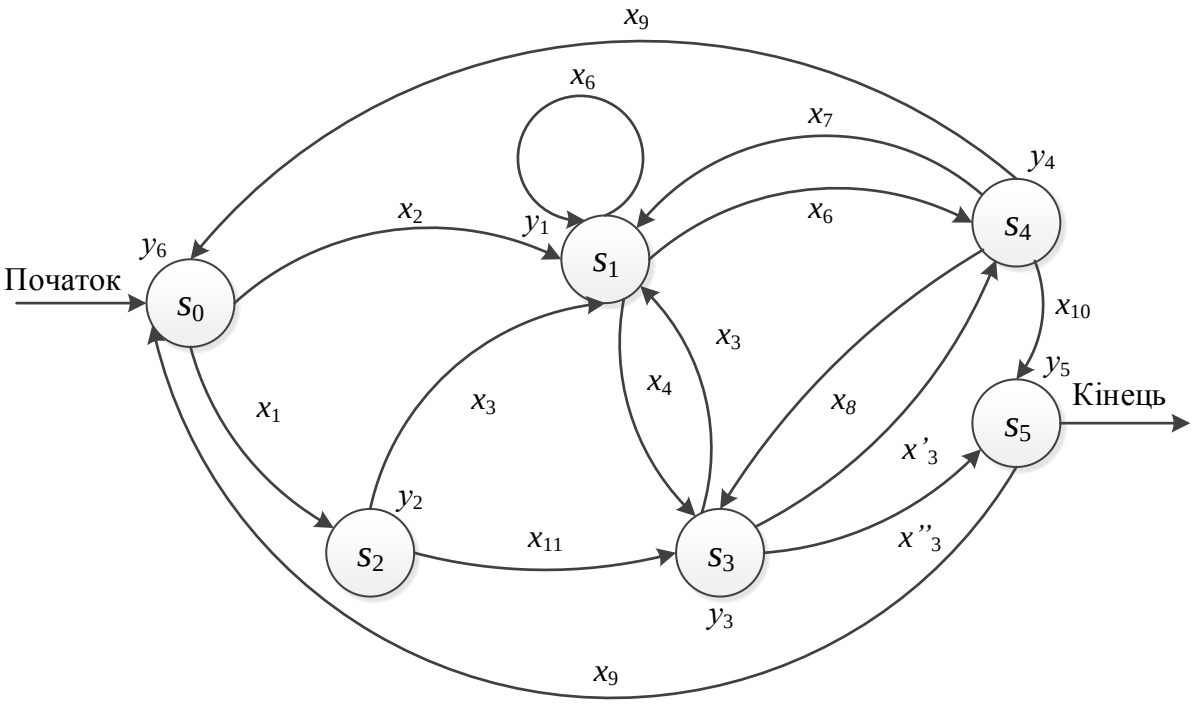

Рис. 2.5. Автомат Мура петлі Бойда з використанням онтологій

Таблиця 2.2. Табличний опис автомату Мура

	y_6	y_1	y_2	y_3	y_4	y_5
	s_0	s_1	s_2	s_3	s_4	s_5
x_1	s_2	-	-	-	-	-
x_2	s_1	-	-	-	-	-
x_3	-	-	s_1	s_4	-	-
x_4	-	s_3	-	-	-	-
x_5	-	s_1	-	-	-	-
x_6	-	s_4	-	-	-	-
x_7	-	-	-	-	s_1	-
x_8	-	-	-	s_4	s_3	-
x_9	-	-	-	-	s_0	s_0
x_{10}	-	-	-	-	s_5	-
x_{11}	-	-	s_3	-	-	-

Автомат містить такі переходи між станами: $\varphi(s_0, x_2) = s_2$, $\varphi(s_2, x_3) = s_1$, $\varphi(s_2, x_{11}) = s_1$, $\varphi(s_2, x_{11}) = s_3$, $\varphi(s_1, x_4) = s_3$, $\varphi(s_1, x_5) = s_1$, $\varphi(s_2, x_6) = s_4$, $\varphi(s_4, x_7) = s_1$, $\varphi(s_4, x_5) = s_0$, $\varphi(s_4, x_8) = s_3$, $\varphi(s_3, x_3) = s_4$ (табл. 2.2).

2.4. Висновки до розділу 2

У другому розділі отримано такі результати:

- розроблено метод моделювання петлі OODA з використанням онтологічного підходу, що на відміну від існуючих підходів дає можливість використовувати об’єктивні знання, подані у вигляді дескриптивної логіки, під час етапів петлі OODA, що у свою чергу, дало змогу розробити математичне забезпечення функціонування СпППР в конкурентному середовищі, де знання є чітко визначеними й документованими;
- подано петлю Бойда, яка використовує онтологію ПО, у вигляді автомата Мура, що на відміну від існуючих підходів дає змогу формалізувати процес підтримки прийняття рішення командирами тактичних ланок, що у свою чергу дало змогу алгоритмізувати функціонування СпППР;
- удосконалено метод цілерозподілу на основі методів штучного інтелекту, а саме генетичних алгоритмів та опрацювання онтологічних знань, який на відміну від існуючих не містить процедур повного перебору, що дало змогу спростити складність алгоритму пошуку ефективного цілерозподілу.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ОСНОВНИХ КОМПОНЕНТ СпППР КОМАНДИРІВ ТАКТИЧНИХ ЛАНОК

У третьому розділі розроблено онтологію СВ ЗСУ, використано модель адаптивної онтології для визначення важливості цілей противника. Онтологія також містить правила поведінки в певних ситуаціях розроблені експертами предметної області, які подані на основі дескриптивної логіки. Розглянето процес проектування СпППР, центральною компонентою якої є розроблена онтологія. СпППР складається із 4 модулів, кожний з яких задає відповідний етап петлі Бойда. Функціонування СпППР відбувається на основі розробленого у розд. 2 автомата Мура. Розроблено алгоритми функціонування окремих модулів СпППР на різних етапах петлі OODA.

3.1. Структурна модель інформаційного простору СВ ЗСУ на основі онтологічного підходу

СВ ЗСУ є системою, яка складається із множини родів військ (механізовані, танкові, аеромобільні війська, ракетні війська і артилерія, війська протиповітряної оборони СВ, армійська авіація СВ); спеціальних військ (військові частини і підрозділи – розвідувальні; інженерні; радіаційного, хімічного, біологічного захисту; зв’язку; радіоелектронної боротьби; інформаційно-психологічних операцій; топогеодезичні; гідрометеорологічні), матеріально-технічного та медичного забезпечення. Організація управління військами полягає в створенні системи управління військами, підтриманні її високої бойової готовності, розвитку і нарощуванні її при веденні бою (операції), а також в підготовці і здійсненні заходів по забезпеченню її безперервної роботи. Тому формування необхідної інформаційної бази для прийняття управлінських рішень важливо як для СВ ЗСУ в цілому так і для окремих військ, тобто підрозділів більш низького рівня [27, 28, 137].

Інформаційне відображення фізичних об’єктів або процесів називають інформаційним об’єктом, а їх сукупність, що інформаційно відображає властивості системи і процеси, що протікають у ній, утворює інформаційний простір [30]. Цей простір складається з інформації, яка містить нормативну, технічну, організаційно-розпорядницьку, звітну документацію; оперативну інформацію про технологічні і організаційні процеси чи інші операції; архівні і статистичні матеріали тощо. Таку інформацію можна розглядати як експліцитні знання в межах предметної області (ПО), а, отже, для її подання та опрацювання найкраще використовувати онтології [16]. Згідно визначення Грубера: «онтологія – це експліцитна специфікація концептуалізації», іншими словами: онтологія – це концептуалізація певної області знань, тобто визначення базових об’єктів (індивідуумів, атрибутів, процесів) предметної області і відношень між ними. Слід зауважити, що основною перевагою такого підходу є те, що онтології стали стандартом інженерії знань, а отже побудований на їх основі інформаційний простір СВ ЗСУ легко інтерпретується та піддається інформаційному опрацюванню існуючими програмними засобами.

З технологічної точки зору, інформаційний простір являє собою сукупність сховищ даних, технологій їхнього ведення і використання, інформаційно-телекомунікаційних систем і мереж, що функціонують на основі єдиних принципів та за загальними правилами, що забезпечує інформаційну взаємодію підрозділів СВ ЗСУ. Застосування найбільш ефективних форм управління СВ ЗСУ пов’язане з активним використанням його інформаційного простору, концепція формування якого являє собою сукупність методів і методик організації інформаційних процесів, що дозволяють здійснити вибір і використання необхідного інформаційно-технічного рішення для синтезу знань про деяку ситуацію, що виникла. Для подання таких знань пропонується використовувати онтології, як спосіб детальної формалізації деякої області знань за допомогою концептуальної схеми.

Розглянемо побудову моделі інформаційного простору СВ ЗСУ та методу його наповнення для підтримки прийняття управлінських рішень [92]. Використання інформаційного простору у системах керування СВ ЗСУ дасть змогу забезпечити органи прийняття рішень доступу до інформації про противника, наявність і рух його військ, надання можливості прискорювати процес обробки інформації і прийняття управлінських рішень.

Формування інформаційного простору СВ ЗСУ вимагає отримання інформації з наперед визначених джерел, її перевірки й аналізу; інформації про прийняті керуючі рішення; формування баз знань, а також розвитку інформаційної системи. Інформаційний простір СВ ЗСУ тісно пов'язаний з інформаційними потоками в системі управління СВ ЗСУ.

Розглянемо взаємодію основних об'єктів СВ ЗСУ. Для цього їх ізоморфно відобразимо на систему множин $\{Q_i\}, i = 1,2,...,n$ і визначимо на цій системі множину відношень між елементами цих множин [14]. Елементи (об'єкти) СВ ЗСУ, наприклад, «Рід військ», «Спеціальні війська», «Матеріально-технічне забезпечення» тощо можна відобразити у вигляді відповідних множин. Таким чином на множині Q визначимо порядок між її елементами, що дозволяє задати модель системи множин у вигляді графу $Q = \langle q_1, q_2, ..., q_n \rangle$. Так Q = «Армійський корпус» складається із окремих механізованих бригад (q_i), танкової бригади (q_j) тощо.

Для керування СВ ЗСУ необхідно забезпечити взаємозв'язок між організаційними і технологічними процесами, з однієї сторони, та інформаційними процесами у системі управління, з іншої сторони. Для забезпечення ефективного управління інформаційний простір повинен формуватись виключно на основі об'єктивної інформації. Для розроблення теоретико-множинної моделі процесу керування СВ ЗСУ застосуємо онтологічний підхід, теорію множин і теорію графів.

Перейдемо до теоретико-множинного моделювання інформаційних процесів СВ ЗСУ. Інформація, що циркулює в об'єктах СВ ЗСУ, формується в

цих об’єктах і передається між ними, відображає переміщення ресурсів, дані про підрозділи, їх укомплектування, техніку тощо. Таким чином відбувається рух потоків відповідної інформації. На графі вказаним потокам інформації відповідають окремі ребра графу. Тобто, взаємодіючи, об’єкти породжують кінцеву множину системних процесів, які, в свою чергу, формують відповідну інформацію у виді відповідних організаційно-розпорядчих документів, параметрів технологічних процесів тощо.

Множина функціональних інформацій формує інформаційні потоки, що направляються в інформаційну систему. Саме на цьому етапі здійснюється збір первинної ресурсно-технологічної інформації.

Для підрозділів СВ ЗСУ кожного рівня інформаційні потоки системи відображають його функціональну та структурну організацію. Рух інформації в таких СВ ЗСУ носить досить складний характер і частково відображає його ієрархічну структуру. В той же час рух слід розуміти не як просте передавання інформації, а як трансформацію інформації з одного стану в інший і формування знань, які потрібні для прийняття управлінських рішень.

Особі яка приймає рішення, для прийняття рішень по управлінню підрозділу СВ ЗСУ необхідна інформація про стан всіх підсистем СВ ЗСУ, що надходить з нижніх рівнів, починаючи від моніторингу СВ ЗСУ і закінчуючи набором альтернативних варіантів розв’язання конкретної задачі. Після прийняття рішення воно реалізується за допомогою організаційних заходів, причому важливим є аналіз прийнятих рішень з метою вироблення рекомендацій покращення управління військами, що є предметом подальших досліджень [122-126].

Для побудови онтологічної моделі, насамперед, необхідно визначити ієрархію понять (множину *C*), СВ ЗСУ. Приклад такої таксономії понять, подану за допомогою діаграми класів UML [111, 127], наведено нижче. Нехай задана множина назв відношень $V = \{v_1, v_2, ..., v_s\}$. Тоді відношення задається як

відображення із C в C, використовуючи елемент множини V: $R: C \xrightarrow{V} C$. Тобто відношення r_i – це триплет вигляду: $r_i = \langle C_{i_1}, v_{i_j}, C_{i_2} \rangle$.

Відношення ієрархії *IS-A* та агрегації є вертикальними, їх позначають R^V. Всі інші відношення є горизонтальними і позначають R^H. Очевидно, що $R^V \cup R^H = R$, $R^V \cap R^H = \varnothing$. Вертикальні відношення задають таксономію понять ПО. Для задання горизонтальних відношень необхідно визначити область визначення відношення (домен) та множину значень (ренг). Наприклад відношенням множини факторів (v_k), які впливають на перебіг ведення бою є: відстань між військами; характеристики ходових властивостей механізованих військ; місцевість (коефіцієнт супротиву руху, видимість цілі); сектор пошуку цілі; розподіл вогню по цілям противника; число необхідних вистрілів для знищення цілі (характеристика розсіювання, захищеність цілі, відстань). В онтологічній моделі таке відношення задається як відображення множини перерахованих факторів (домен C_i) на концепт «Перебіг ведення бою» (ранг C_j), тобто $R: C_i \xrightarrow{v_k} C_j$.

Інакше, кажучи, множина відношень R задає відображення деякої множини X (область визначення) в множину Y (множина значень), тобто $R: X \to Y$. Наведемо ряд прикладів такого відображення з військової тематики.

1. Проводиться розвідка на уточнення координат цілі. Показником ефективності може бути збільшення ймовірності враження даної цілі.

X – ціль виявлена достатньо точно, Y – операція відбулась успішно.

X – ракета попадає не дальше як 500м від цілі, Y – ціль вражено.

2. Проводиться розвідка на уточнення рішення (якщо ціль вражена стрільба зупиняється, якщо ні – продовжується до використання всіх боєприпасів). Завдання – попередити зайві витрати боєприпасів на ціль, яка вже вражена. В якості показника ефективності може бути середнє число з економлених боєприпасів (ракет).

X – ціль вражена, Y – стрільба зупиняється.

3. Проводиться напад на противника. Важливими факторами успішної операції є:

X – стрільба з літака по літаку, *Y* – тип цілі, дальність стрільби, швидкість цілі, швидкість стріляючого літака.

X – бомбардувальний наліт, *Y* – вид і розміри цілі, число літаків, що приймають участь у нападі, дальність стрільби, висота польоту, спосіб бомбометання, наявність чи відсутність радіоперешкод.

X – бомбардувальний наліт на деяку територію (площу), *Y* – час виконання операції, вартість затрачених ресурсів, кількість обслуговуючого персоналу, середня глибина проникнення противника на територію, що охороняється.

4. Вибір способу ведення стрільби:

X – ціль малорозмірна (одиночна ціль), *Y* – зосереджена стрільба.

X – ціль майданна, *Y* – рознесена стрільба.

5. Щоб оцінити ефективність стрільби, крім характеристик розсіювання (точність пострілів), потрібно враховувати характеристики вражаючої дії боєприпасів по цілі (конструкція, вага і руйнівна міць снаряду, конструкція і міцність цілі).

X – снаряд попав у точку з деякими координатами, *Y* – ціль знешкоджено.

Функції інтерпретації *F* задають обмеження або приймання певних значень властивостями концептів. Приклад таких функцій наведено на рис. 3.1, які задають певні значення, що приймають основні характеристики ВМР-1.

Розроблена нами онтологія СВ ЗСУ на даний момент містить 384 понять, 207 відношень, 27 % понять є визначеними. Поняттями предметної області є: бойові машини, гармати, артилерійські снаряди тощо. Відношеннями між поняттями є: «має максимальну швидкість», «має гармату» тощо. Приклади окремих понять та відношень наведено на рис. 3.1. В онтології містяться дані з нормативних таблиць в яких відображені коефіцієнти (переведення з одних

величин в інші, ефективності вогневих засобів тощо), різні нормативні розрахункові величини (ймовірності ураження цілей в залежності від відстані та вогневого засобу, кількість боєкомплекту тощо), усереднені нормативи виконання окремих дій підрозділів, вогневі можливості артилерійських підрозділів тощо. Конкретні екземпляри понять зберігаються у БД. Це дає змогу командирам тактичних ланок відслідковувати стан власних підрозділів.

За допомогою процедур інтелектуального аналізу (*IA*) змісту онтології отримуємо метазнання, які можна використати як знання *U* для управління підрозділами СВ ЗСУ. Таким чином отримаємо таку структурну модель автоматизованої системи управління СВ ЗСУ:

$$S = \langle O, IA, U \rangle .$$

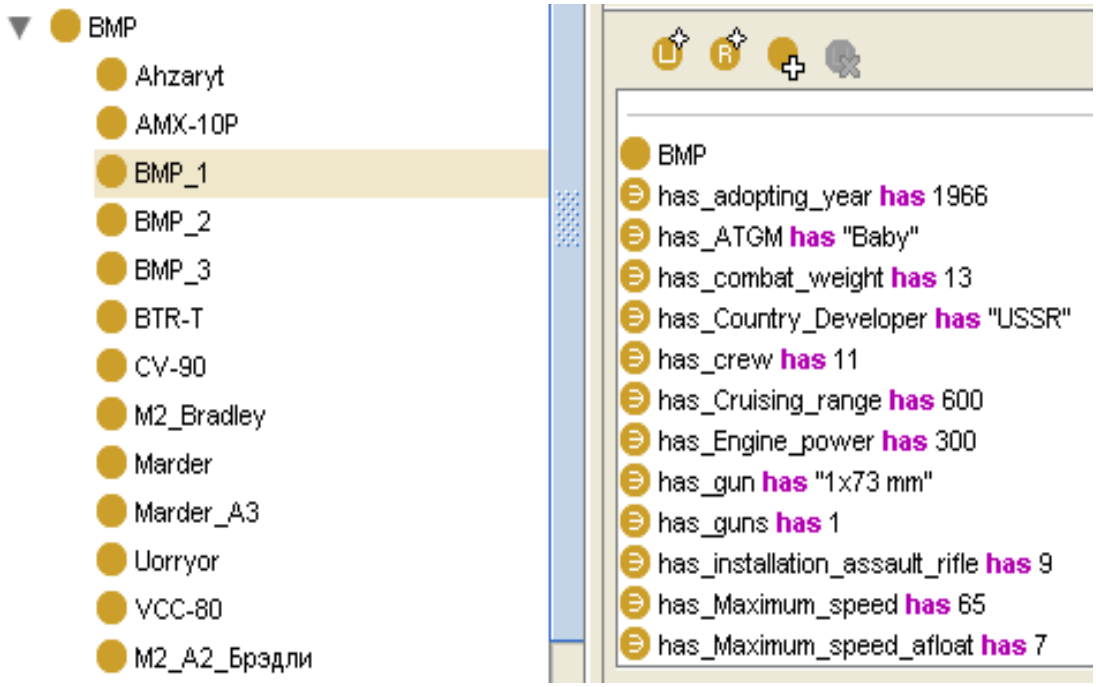

Рис. 3.1. Приклад функцій інтерпретацій онтології СВ ЗСУ

Для підвищення ефективності можливих рішень в онтології подані знання експертів (генералів, полковників СВ ЗСУ тощо) щодо поведінки в певних ситуаціях за допомогою SWRL-правил. Ці правила записані на мові дескриптивної логіки (DL). Наприклад експертне правило «накрити вогнем нашої артилерії ПЗРК противника під час висадки нашого десанту з вертольота

на територію ?x, якщо відстань до ПЗРК противника менша-рівна ?y» на мові DL (як опис українською мовою) в нашій онтології подано у такому вигляді: (Landing (Desant, ?x)) ∧ (Location (PZRK enemy, ?x) ≤ ?y) → Cover (our artillery, PZRK enemy).

Щоб використати процедури інтелектуального аналізу необхідно промоделювати перебіг воєнних дій за допомогою запропонованої моделі інформаційного простору. Таке імітаційне моделювання є подальшим етапом наукових досліджень.

Отже побудовано формальну структурну модель інформаційного простору СВ ЗСУ на основі онтологічного підходу, що дає змогу досліджувати та аналізувати як структуру СВ ЗСУ так і ефективність задач, які розв’язують її окремі підрозділи. Ядром такого інформаційного простору є онтологія, яка задає експліцитні знання (нормативні документи, ієрархія військ, тактико-технічні показники бойових машин тощо). Мета побудови онтології полягає в його аналізі, ціль якого, у свою чергу, полягає у виробленні рекомендацій покращення боєздатності (управління) СВ ЗСУ. Пропонована модель інформаційного простору слугуватиме базою для розроблення методики покращення боєздатності СВ ЗСУ.

3.2. Побудова онтології СВ ЗСУ

3.2.1. Інструменти для інженерії онтологій

Оцінка функціональності програм для побудови онтологій залежить від:

- практичної задачі (цілі розробника);
- області знань, в рамках якої будується онтологія;
- онтології, що розробляється.

Для онтології важливим критерієм є можливість перенесення її на інші платформи, можливість перевести її на інші формальні мови. Так, засіб створення онтологій Protege [93, 130] – вільно розповсюджувана локальна Java програма на базі Windows, дозволяє виконувати експорт в RDF, RDFS, XML,

HTML, OWL, Clips, N3, TURTLE. Програма призначається для побудови (створення, редагування та перегляду) онтологій моделей прикладної області. Її ціль – допомогти розробникам програмного забезпечення у створенні і підтримці явних моделей предметної області і у включенні цих моделей безпосередньо в код програми. Protege складається з трьох головних частин, що повинні використовуватися по черзі. Спочатку йде редактор онтологій, що дозволяє проектувати онтології розвертаючи ієрархічну структуру і включаючи абстрактні або конкретні класи і слоти. Грунтуючись на сформованій онтології, Protege здатний генерувати інструмент придбання знань для введення екземплярів онтології. Остання частина програми – інтерпретатор схем, який дозволяє робити екземпляри для класів і підкласів. Інструмент має повний графічний інтерфейс, що є дуже зручним для використання недосвідченими користувачами.

Для досягнення поставленої мети використано: теорію множин та методи подання знань для моделювання структури онтології та розроблення процедур опрацювання онтологічних знань; теорію дослідження операцій, ймовірностей та штучного інтелекту для розроблення функціональності окремих модулів СпППР; методи системного аналізу, методи об’єктно-орієнтованого аналізу і проектування – для розроблення архітектури СпППР; теорію реляційних баз даних, методи штучного інтелекту, об’єктно-орієнтоване програмування – для програмної реалізації розроблених моделей, методів та алгоритмів функціонування окремих модулів СпППР .

3.2.2. Структура класів онтології

У процесі програмної реалізації на основі інформації про ЗСУ було розроблено ієрархічну структуру класів онтології [84-87] (рис. 3.2) спроектовану в Protege, який є гнучким, незалежне від платформи середовищем для створення і редагування онтологій та баз знань. Порожня онтологія містить лише один клас owl: Thing – цей клас відображає множину всіх об’єктів,

оскільки всі наступні класи є його підкласами. Це системний клас, який існує за замовчуванням і в процесі створення онтології не змінюється.

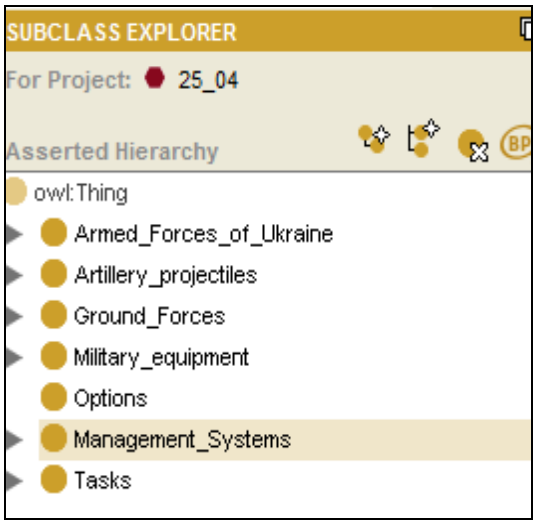

Рис. 3.2. Ієрархія класів першого рівня

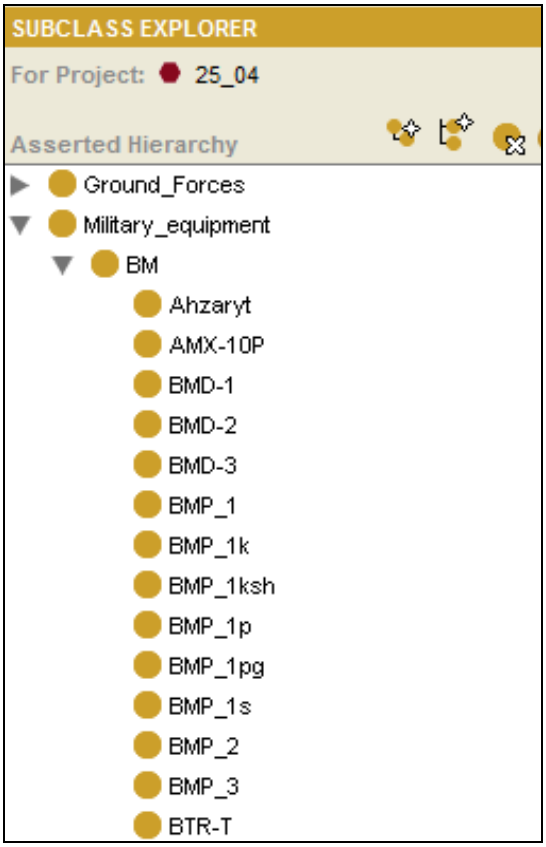

Рис. 3.3. Ієрархія класу Military equipment (військове обладнання)

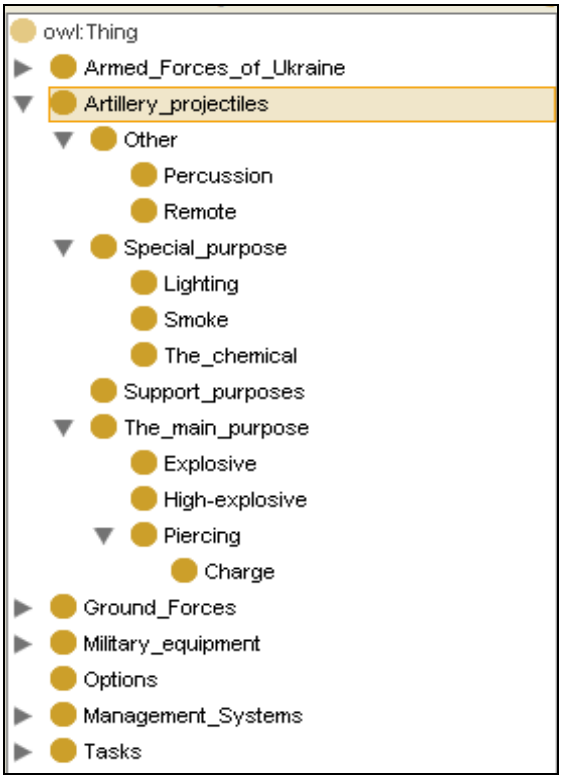

Рис. 3.4. Ієрархія класу Artillery projectiles (артилерійські снаряди)

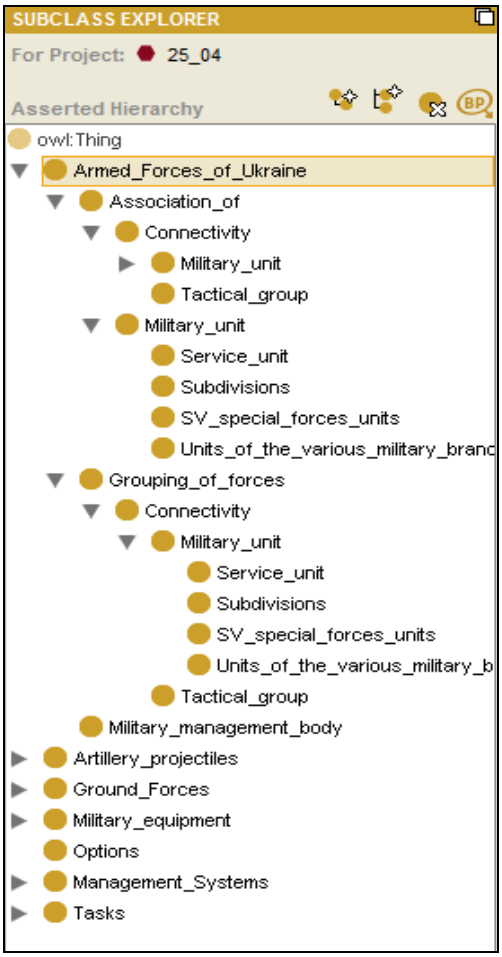

Рис. 3.5. Ієрархія класу Armed Forces of Ukraine (ЗСУ)

Ієрархія класу Military equipment (військове обладнання) подана на рис. 3.3.

При створені усіх підкласів класу вказуємо, що вони розділені (disjoint), тобто об’єкти одного з них не можуть бути об’єктами іншого. Для цього існує вікно «Disjoints Widget» у правому нижньому куті закладки «OWLClasses». На рис. 3.4. та рис. 3.5. шляхом додавання класів з ієрархії, вибрано ті класи, які було розділено (рис. 3.6).

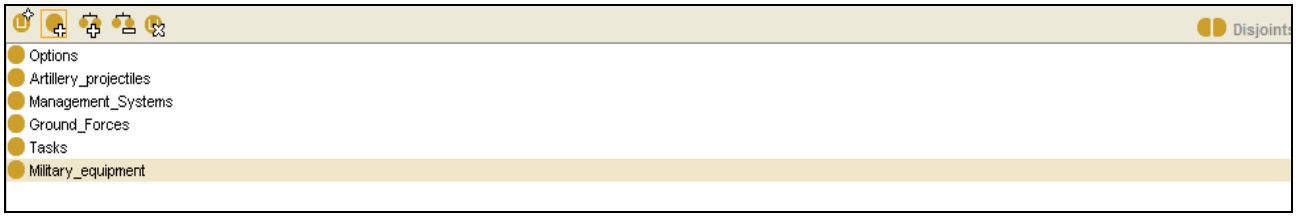

Рис. 3.6. Вікно розділення класів

Для класу може бути визначеною множина умов, яким має відповідати клас. Для створення нової властивості використовується закладка «Properties» головного вікна програми. Вікно створення властивостей складається із двох частин – Списку властивостей (Property Browser) та Редактора властивостей (Property Editor). На рис. 3.7 створено властивість has_gun для класів BM (домен) та GUN (ранг).

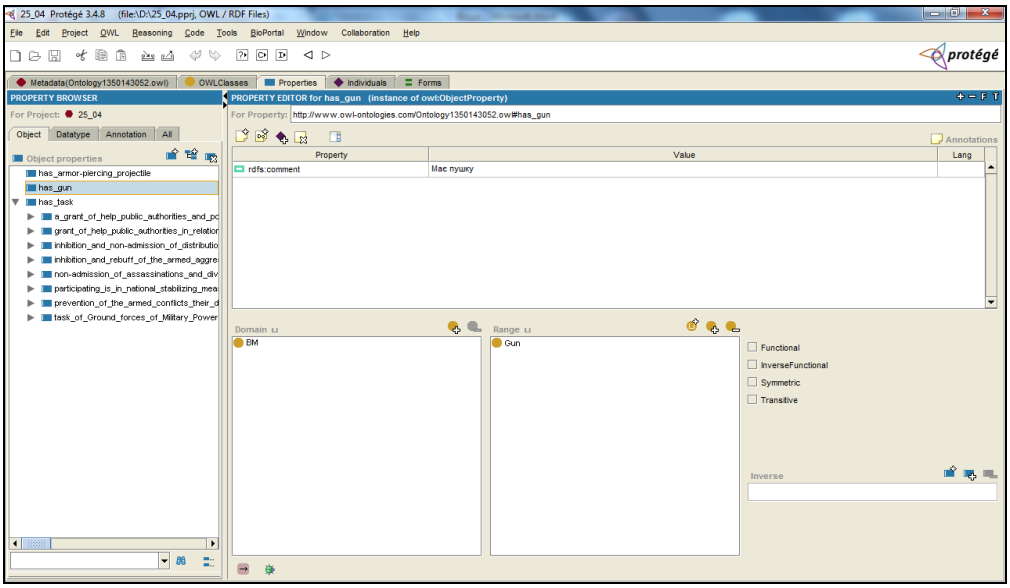

Рис. 3.7. Вікно властивостей

3.2.3. Використання дескриптивної логіки для подання експертних знань в онтології СВ ЗСУ

Дескриптивна логіка (англ. Description logics, DL, іноді її ще називають описовою логікою) [13] – сімейство мов представлення знань, що дають змогу описувати поняття предметної області у формалізованому вигляді.

Вони поєднують у собі, з одного боку, логічні вирази, а з іншого – обчислювальні властивості, такі як розв'язність і відносно невисока обчислювальна складність, що дає змогу легко їх застосовувати на практиці. Тобто, дескриптивна логіка, є компромісом між записом логіки формалізованими виразами і їх розв'язністю (логічне виведення нових знань). Дескриптивну логіку можна розглядати як розв'язні фрагменти логіки предикатів, синтаксично ж вони близькі до модальних логік.

Свою сучасну назву дескриптивна логіка отримала в кінці минулого століття. Колишні назви були такі (у хронологічному порядку): термінологічні системи, логіки концептів. Спочатку дескриптивна логіка зародилися як розширення фреймових структур та семантичних мереж механізмами формальної логіки. У даний час вона є важливою в концепції семантичної павутини, де її використовують під час побудови онтологій. Фрагменти OWL-DL та OWL-Lite мови веб-онтологій OWL також базуються на дескриптивній логіці.

Дескриптивна логіка оперує поняттями концепт і роль, «одномісний предикат» (або множина, клас) та «двомісний предикат» (або бінарне відношення). Інтуїтивно, концепти використовуються для опису класів деяких об'єктів, наприклад, «Особи», «Солдати», «Бойові машини». Ролі використовуються для опису двомісних відношень між об'єктами, наприклад, на множині осіб є двомісне відношення «X є_начальником_для Y», а між особами і бойовими машинами є двомісне відношення «X водить Y», де в якості X і Y можна підставляти конкретні об'єкти. За допомогою мови DL можна формулювати твердження загального вигляду – про класи взагалі

(будь-який Солдат є Особа, будь-яка Бойова_Машина належить не більше, ніж до одного підрозділу) та приватного виду – про конкретні об'єкти (Іваненко є солдат, Петренко водить БМП).

Мовою дескриптивної логіки є набір тверджень загального вигляду називається TBox, набір тверджень приватного виду – ABox, а разом вони складають так звану БЗ, або онтологію. Як тільки онтологія побудована, можна отримувати знання, які виводяться з наявних в онтології знань. Для цього використовуються прикладні програмні системи (reasoners), які дають змогу автоматизовано виводити знання з онтологій та виконувати інші операції з онтологіями. Ми використали такий програмний засіб як Pellet [2].

Для підвищення ефективності можливих рішень в онтології подані знання експертів щодо поведінки в певних ситуаціях за допомогою SWRL-правил. Ці правила записані на мові дескриптивної логіки. Наприклад, експертне правило «накрити вогнем нашої артилерії ПЗРК противника під час висадки нашого десанту з вертольота на територію ?x, якщо відстань до ПЗРК противника менша-рівна ?y» на мові DL в нашій онтології подано у такому вигляді: (Landing (Desant, ?x)) ∧ (Location (PZRK enemy, ?x) ≤ ?y) → Straddle (our artillery, PZRK enemy).

3.2.4. Використання адаптивної онтології для оцінювання важливості цілей противника

Для визначення важливості цілей противника використано модель адаптивної онтології, розробив д.т.н., професором В. В. Литвин [41-43, 97, 100, 114, 69].

Адаптивна онтологія визначається як:

$$\hat{O} = \left\langle \hat{C}, \hat{R}, F \right\rangle, \tag{3.1}$$

де $\hat{C} = \langle C, W \rangle$, $\hat{R} = \langle R, L \rangle$, де у свою чергу W – важливість понять C, L – важливість відношень R.

Для задання W у роботі [34-37] запропоновано такі методи:

- за рахунок експертних оцінок;
- частотний метод (частота зустрічання термінів у науковій літературі);
- ймовірнісний метод (адаптивна онтологія подібна до байєсівської мережі);
- за рахунок аналізу (статистичного, інтелектуального) інформаційних джерел, які описують ПО, в якій функціонує ІА.

Для задання важливості цілі використаєм експертні оцінки. Важливість цілі визначається шкодою, яку ми завдаємо противнику, знищивши цю ціль. Для градації цілей проведено опитування експертів військової галузі для оцінки важливості цілі за 10 - бальною шкалою (1 – важливість цілі кулемет, 10 – важливість цілі КП бригади). Важливість елемента онтології, який задає ціль противника, визначається як середнє арифметичне експертних оцінок, тобто $W \in [1,10]$. Вагу цих цілей надалі використаєм під час побудови алгоритму імітаційного моделювання перебігу бою та цілерозподілу.

Важливість елемента онтології, який задає ціль противника, визначається як середнє арифметичне експертних оцінок, тобто $W \in [1,10]$. Тоді ціль противника з максимальною вагою, як елемент онтології, визначається за формулою:

$$C_{Z^*} = \arg\max_{C_Z} \left(\sum_{\tilde{C}_i \to C_Z} W_{\tilde{C}_i} + W_{C_Z} \right). \qquad (3.2)$$

Також цю формулу використовуємо під час цілерозподілу, якщо кількість засобів ураження набагато менша від кількості цілей $(n \prec N)$, ранжуючи наперед цілі.

3.3. Проектування СпППР

Існуючі досягнення в області нових інформаційних технологій відкривають величезні можливості для вирішення ключових проблем у ЗСУ. До числа таких проблем можна віднести визначення перспективних систем

озброєння для видів збройних сил; вироблення рекомендацій з оптимального складу збройних сил і родів військ; ефективний розподіл матеріальних та грошових ресурсів при розробці перспективних систем озброєння. Крім того, потужні ЕОМ, підключення в розподілену комп'ютерну мережу, можуть дозволити вирішувати інші важливі проблеми в області військової науки. Зокрема, визначення раціональних варіантів і способів ведення сучасних війн, недопущення стратегічних прорахунків, що мали місце у другій світовій війні (особливо, в її початковий період), виявлення можливих варіантів ведення війни із застосуванням звичайного озброєння, здійснення практичної підготовки вищого керівного та ін.

Математичні моделі, реалізовані на засобах нових інформаційних технологій, дозволяють змістити рішення військових проблем у практичну площину. За допомогою математичних моделей можна віртуально відтворити всі особливості військових дій і на науковій основі (за допомогою кількісних оцінок ефективності) знайти обґрунтоване рішення військових проблем.

Основні задачі системи, що проектується в цій роботі:

- «відтворювати» (імітувати) застосування всіх засобів і систем озброєння протиборчих сторін, задіяних у планованих операціях (бойових діях). Враховуючи, що імітовані засоби озброєння складають основу армій, бригад, підрозділів воюючих сторін, то в ході моделювання імітуються дії всіх цих військових формувань;
- детальний імітування в моделях всіх процесів військових дій, включаючи жорстке протистояння противника, відкриває практично необмежені можливості у вивченні впливу численних факторів на хід і результат планованих операції (бойових дій);
- забезпечити наочність віртуальних боїв, битв і операцій. Враховуючи, що імітаційні моделі двосторонні, то на екрані монітора відображаються дії своїх угруповань військ та угруповань військ противника в часі і в просторі. причому, все це відображається на тлі цифрової карти місцевості, або в

панорамному зображенні. Користувачі моделей можуть візуально спостерігати за модельованими операціями (бойовими діями) воюючих сторін і, тим самим, більш детально вивчити вплив численних факторів на хід і результат військових дій.

3.3.1. Діаграма станів

Головне призначення діаграми станів – описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу. Найчастіше діаграми станів використовуються для опису поведінки окремих екземплярів класів (об’єктів), але вони також можуть бути застосовані для специфікації функціональності інших компонентів моделей, таких як варіанти використання, актори, підсистеми, операції та методи.

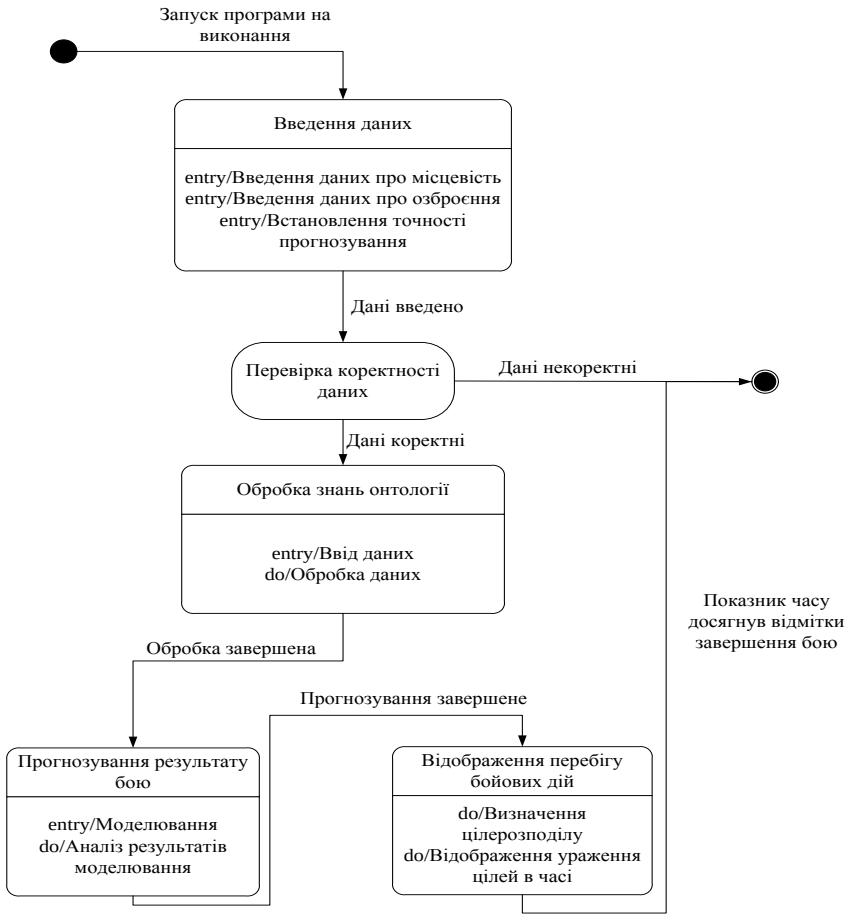

Рис. 3.8. Діаграма станів

На рис. 3.8 зображено діаграму станів проектованої системи. На діаграмі відображено, що після початкового стану відбувається перехід в стан введення даних. Після введення даних система переходить у стан перевірки даних. Якщо вхідні дані некоректні, то наступним буде кінцевий стан, інакше система спочатку перейде в стан роботи з онтологією, після цього в стан прогнозування, надалі в стан моделювання перебігу бойових дій і тільки після того в кінцевий стан.

3.3.2. Діаграма діяльності

При моделюванні поведінки проектованої або аналізованої системи виникає необхідність не тільки представити процес зміни її станів, але і деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій. Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності.

На рис. 3.9 зображено діаграму активності для розробленої системи. З діаграми можна побачити, що в процесі взаємодії системи з користувачем існуватиме три доріжки:

- користувач,
- система підтримки прийняття рішень,
- онтологія.

Спочатку користувач виконує такі дії:

- вводить дані про карту (розміри поля бою, рельєф, замаскованість, прохідність);
- вводить дані про бойову техніку (кількість бойових одиниць, їх тип та координати);
- задає точність моделювання (кількість ітерацій, крок моделювання).

В результаті цих дій формуються вхідні дані. Наступною дією система перевіряє вхідні дані. Якщо вони неправильні виводиться повідомлення про помилку, інакше на основі вхідних даних і знань в онтології робиться прогноз

про перебіг бою та відбувається моделювання бойових дій між «червоними» та «синіми», результати виводяться на екран. Після цього система переходить у кінцевий стан.

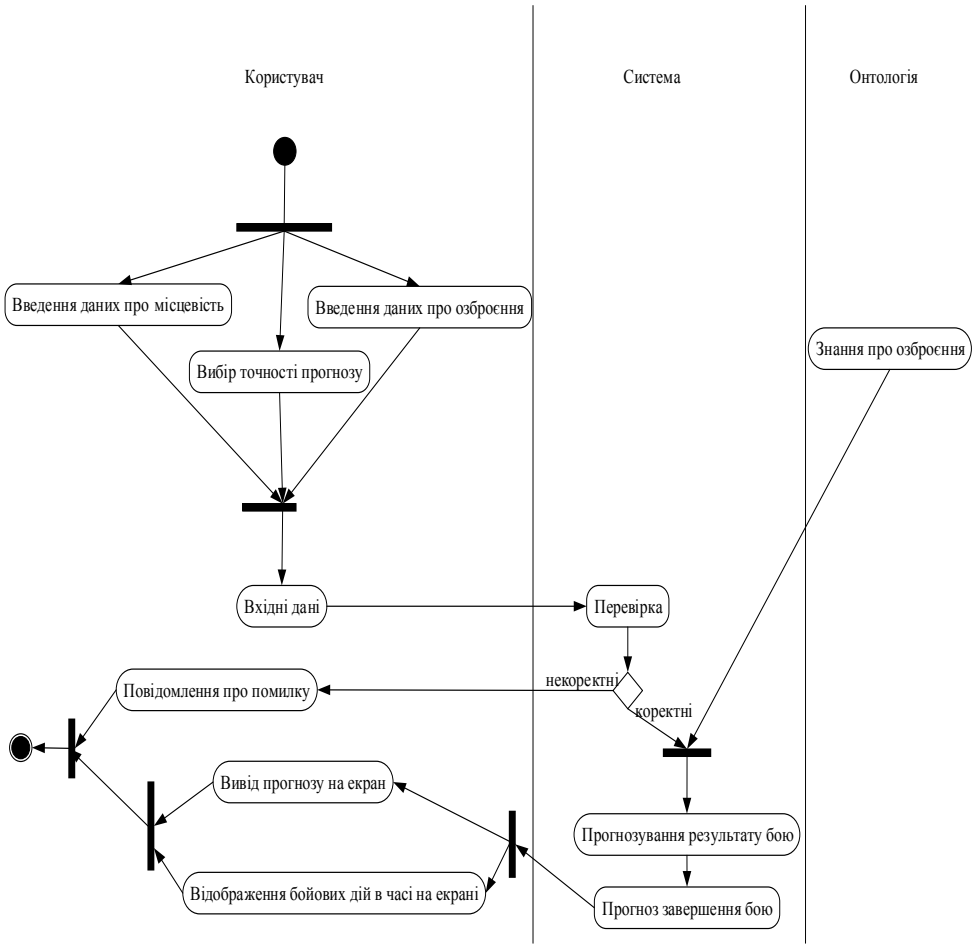

Рис. 3.9. Діаграма діяльності

3.3.3. Діаграма класів

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відображати, зокрема, різні взаємозв'язки між окремою сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру і типи відношень. На цій діаграмі не вказується інформація про тимчасові аспекти функціонування системи. З цієї

точки зору діаграма класів є подальшим розвитком концептуальної моделі проектованої системи.

Для даної предметної області побудоване відношення узагальнення. Відношення узагальнення є звичайним таксономічним відношенням між загальнішим елементом (батьком або предком) і частковішим, або спеціальним елементом (нащадком). Таке відношення може використовуватися для представлення взаємозв'язків між пакетами, класами, варіантами використання й іншими елементами мови UML.

На рис. 3.10 – 3.19 зображено діаграми класів деяких об'єктів предметної області.

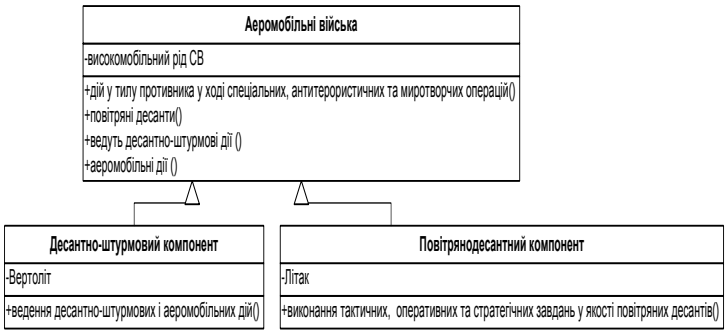

Рис. 3.10. Діаграма класів з батьківським класом «Аеромобільні війська»

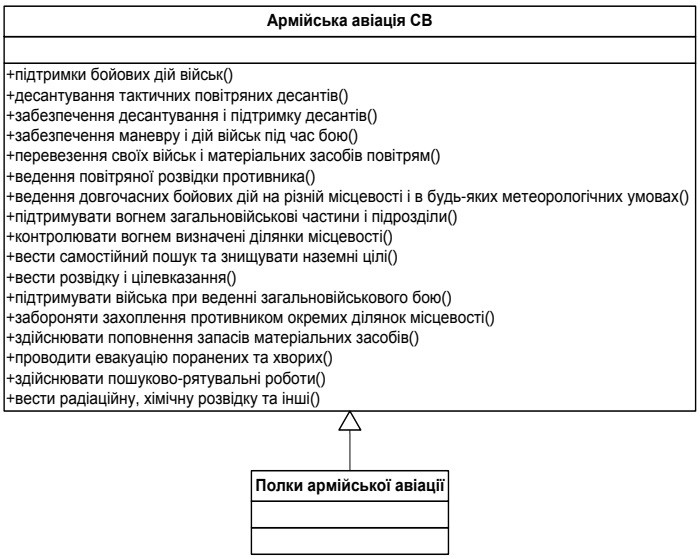

Рис. 3.11. Діаграма класів з батьківським класом «Армійська авіація СВ»

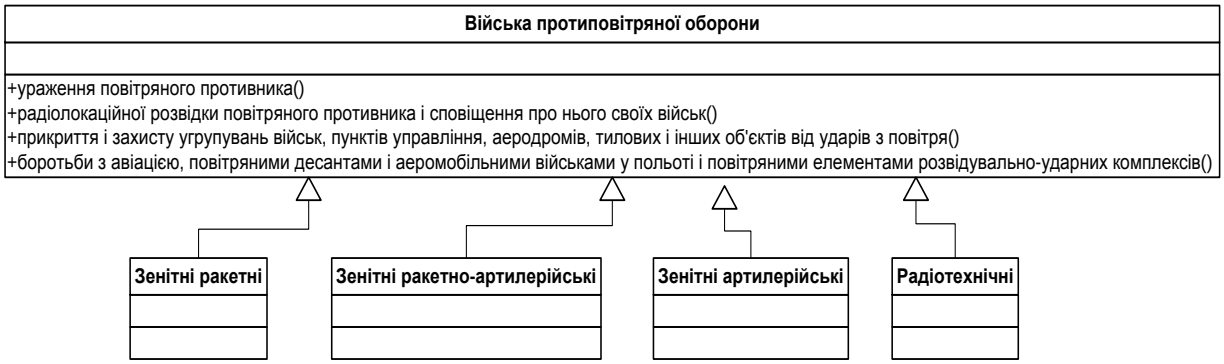

Рис. 3.12. Діаграма класів з батьківським класом «Війська протиповітряної оборони».

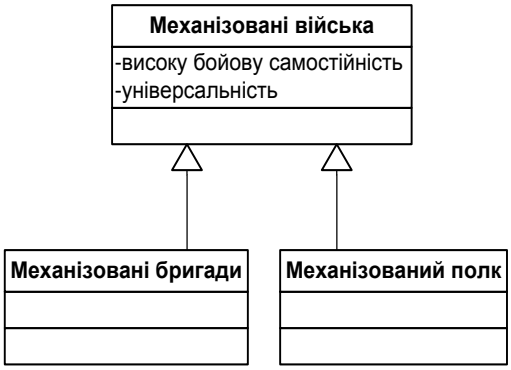

Рис. 3.13. Діаграма класів з батьківським класом «Механізовані війська»

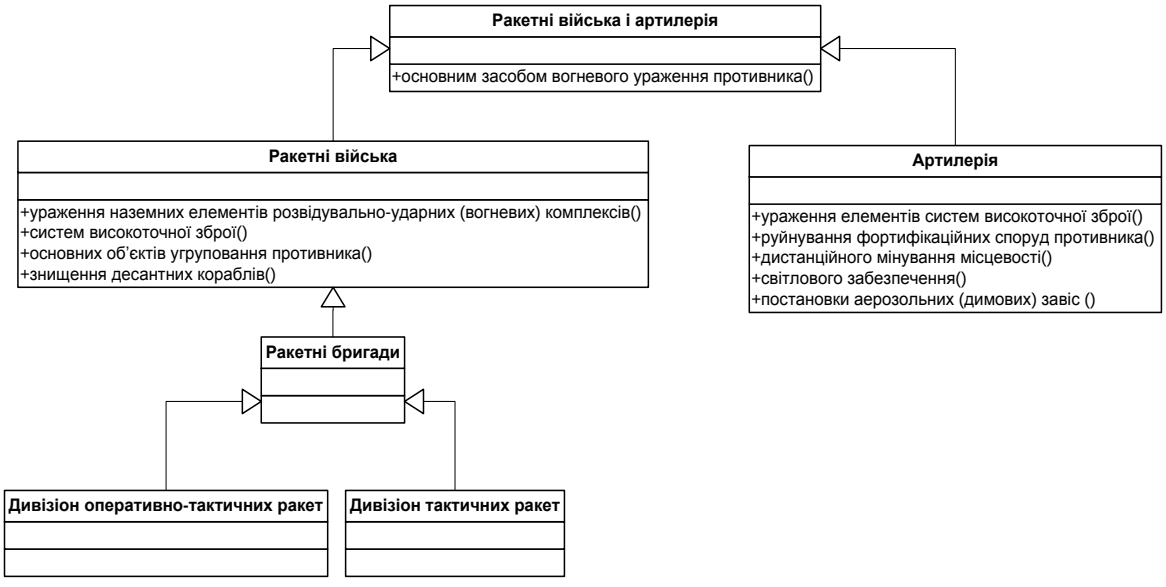

Рис. 3.14. Діаграма класів з батьківським класом «Ракетні війська і артилерія»

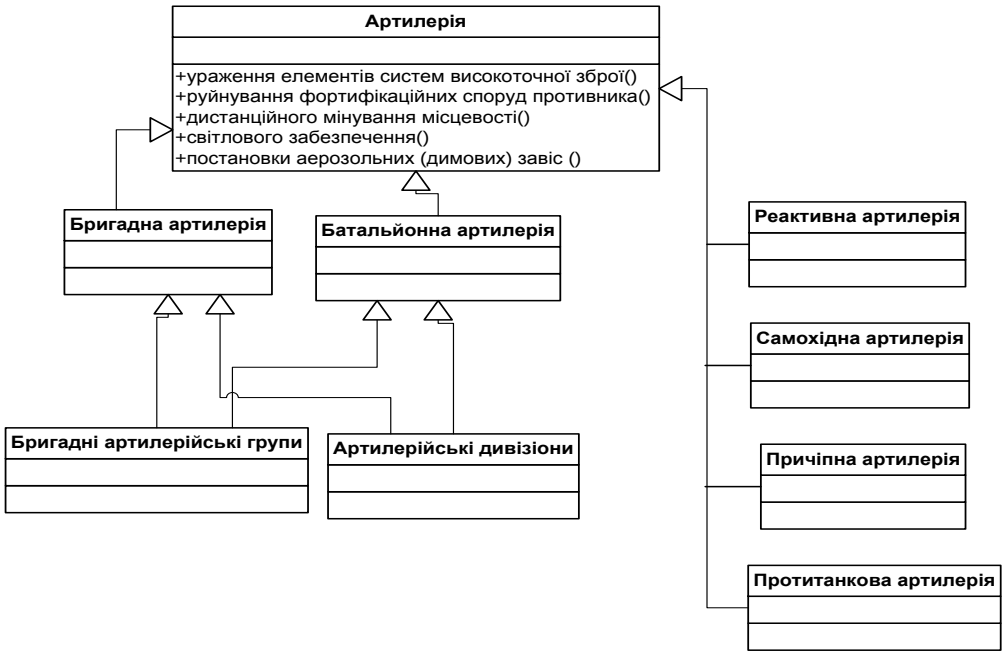

Рис. 3.15. Діаграма класів з батьківським класом «Артилерія»

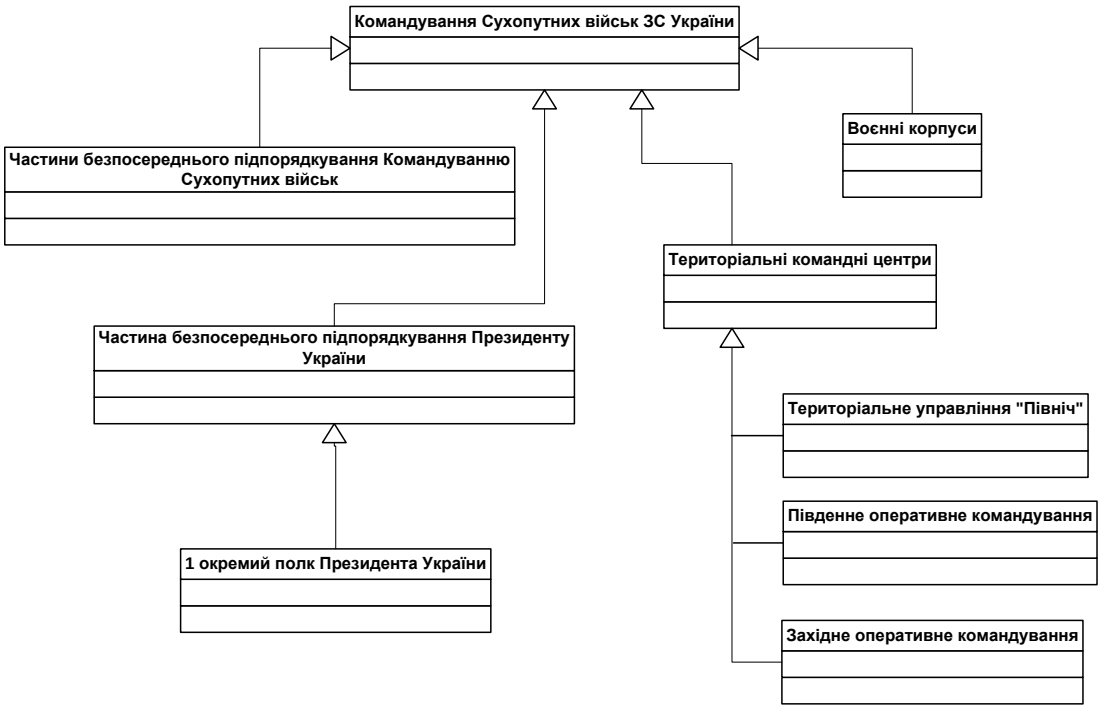

Рис. 3.16. Діаграма класів з батьківським класом «Командування СВ ЗСУ»

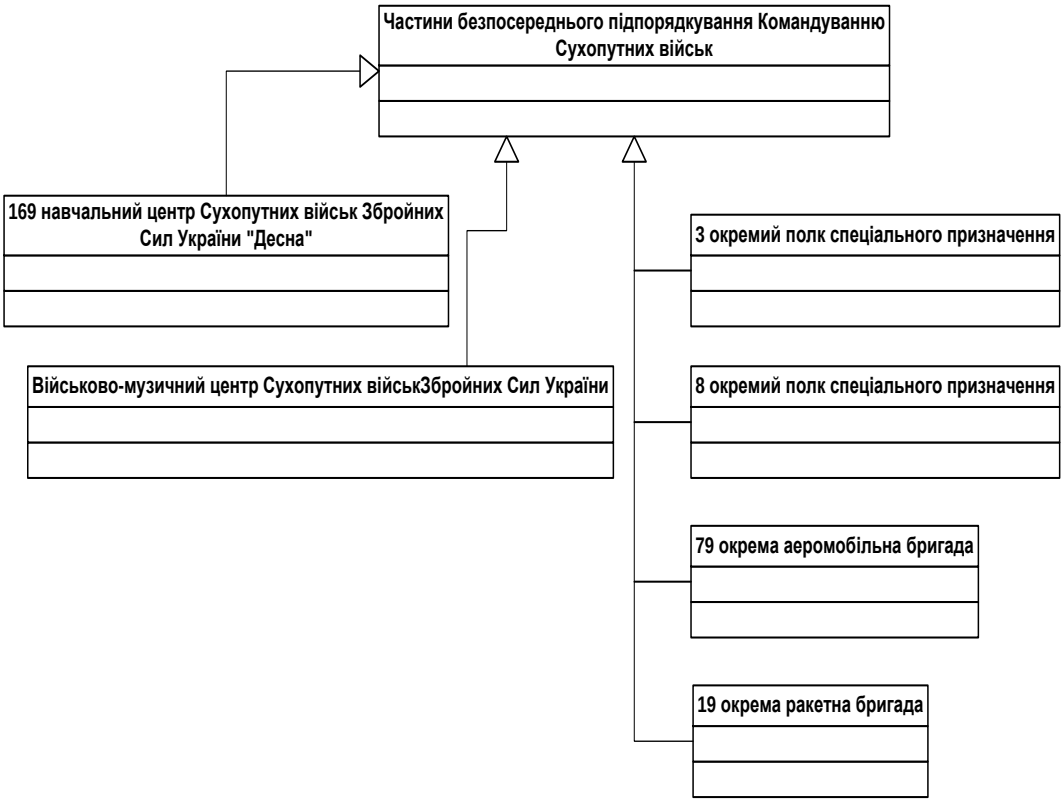

Рис. 3.17. Діаграма класів з батьківським класом «Частини безпосереднього підпорядкування командуванню СВ»

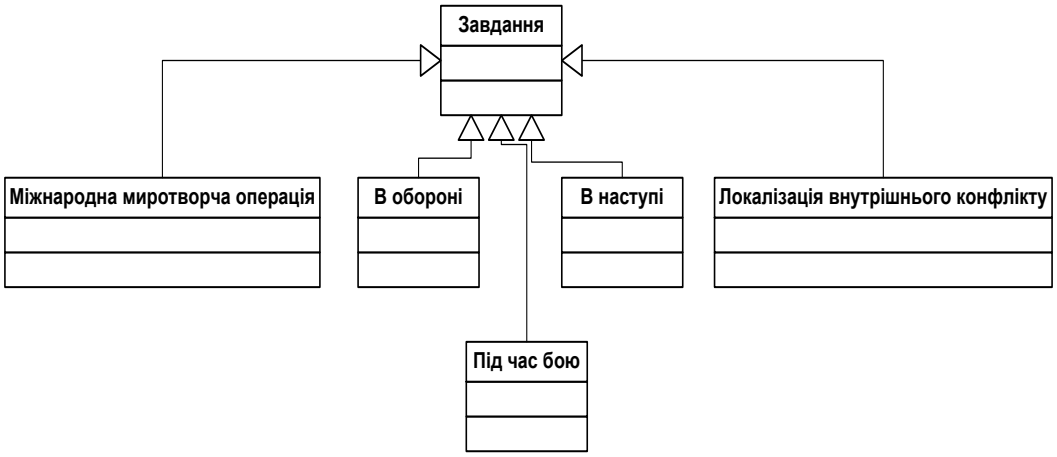

Рис. 3.18. Діаграма класів з батьківським класом «Завдання»

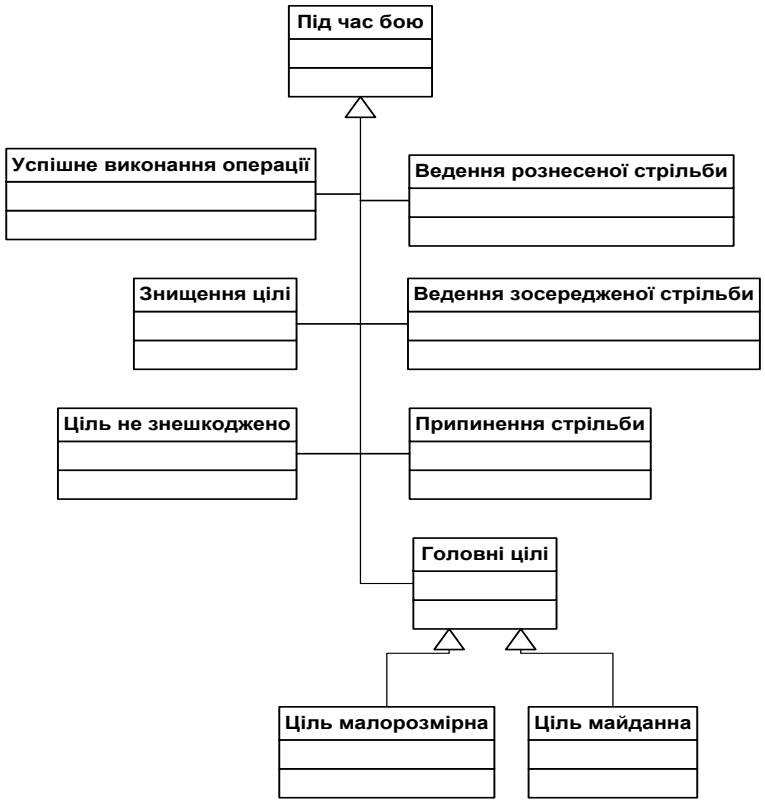

Рис. 3.19. Діаграма класів з батьківським класом «Під час бою»

3.3.4. Діаграма варіантів використання

На діаграмах варіантів використання відображається взаємодія між варіантами використання, що представляють функції системи, і діючими особами, що представляють людей або системи, які отримують або передають інформацію в дану систему. З діаграм варіантів використання можна отримати досить багато інформації про систему. Цей тип діаграм описує загальну функціональність системи.

На рис. 3.20. зображено діаграму використання для даної предметної області. На ній можна побачити, що основне призначення системи – аналіз бою, що включає прогноз результату, то пошук оптимальної позиції розміщення військ. Акторами в даному випадку буде користувач і онтологія, оскільки вони знаходяться поза системою.

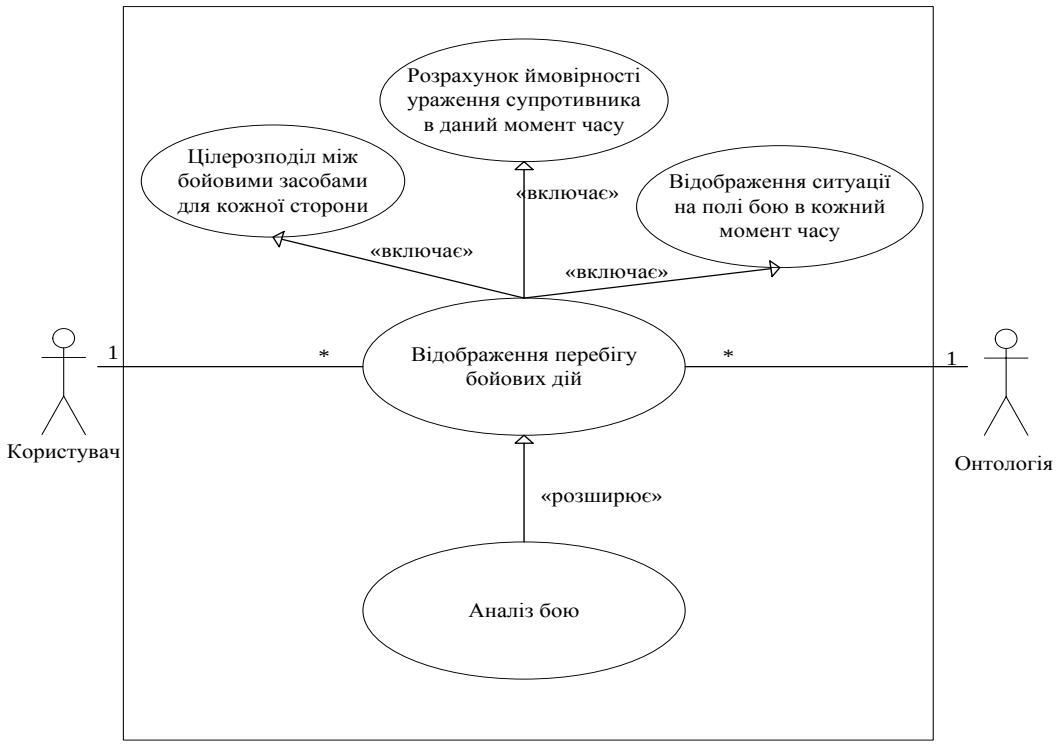

Рис. 3.20. Діаграма варіантів використання

3.4. Побудова алгоритмів функціонування модулів

У дисертаційній роботі розроблено математичне забезпечення для кожного з етапів петлі OODA при їх взаємодії з онтологією, а також побудовано алгоритми функціонування модулів розвідування, імітаційного моделювання та цілерозподілу на основі генетичного алгоритму, які входять у склад СпППР.

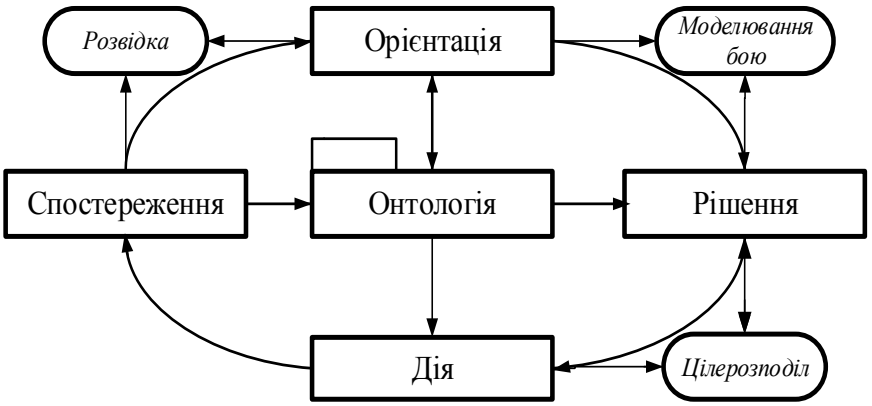

Рис. 3.21. Етапи петлі OODA та їх взаємодія з онтологією

Область військових технологій характеризується відсутністю нормативно встановлених визначень і строгої класифікації технологій. Військові технології постійно розвиваються, що відбивається у розширенні та зміні понятійної системи .

Побудова формальної онтології, що включає аксіоматичну складову, для такої предметної області є надзвичайно складним завданням. Сфера військових технологій являє собою складно-структуровану область, що включає як абстрактні, узагальнюючі поняття, так і прикладну термінологію, яка містить поняття по конкретних реалізаціям військових технологій. У структурі онтологічної моделі військових технологій пропонується виділити чотири основних рівні. Перший рівень – онтологія подання знань. Мета першого рівня – створення мови для специфікації онтологій більш низьких рівнів. Оскільки область військових технологій є підкласом області технологій, то була введена відповідна онтологія верхнього рівня. Онтологія верхнього рівня може використовуватися як основа для побудови онтологій різних предметних областей. Вона описує основні поняття в галузі технологій, такі як «технологія», «знання», «технофакт», «технологія продукції», «виробнича технологія», «подвійна технологія», «виріб» та ін.

Третій рівень містить онтологію предметної області - військових технологій. До основних концептів області військових технології належать: «військова технологія», «технологія озброєння», «технологія виробництва озброєння», «базова військова технологія», «критична військова технологія», «перелік базових та критичних військових технологій», «програма розвитку базових військових технологій».

Прикладні онтології військових технологій, складові четвертого рівня, описують множину реалізацій військових технологій. Вони містять специфічну інформацію - концепти і відносини, що розкривають особливості певних видів зброї і військової техніки (лазерна зброя, динамічний захист, гіперзвуковий

носій, комп’ютер на основі перепрограмованих логічних матриць, система навігації та ін.)

3.4.1. Розвідка та передача даних

Для вдосконалення етапу спостереження розроблено мобільний застосунок «Military intelligence» розроблений для швидкого та ефективного здійснення військової розвідки. Він призначений для збору та узагальнення відомостей про бойовий склад, положення, стан угруповань військ наземного противника, характер його дій і намірів, сильних та слабких сторін, а також ступінь та характер інженерного обладнання.

Алгоритм роботи додатку зображено на рис. 3.22.

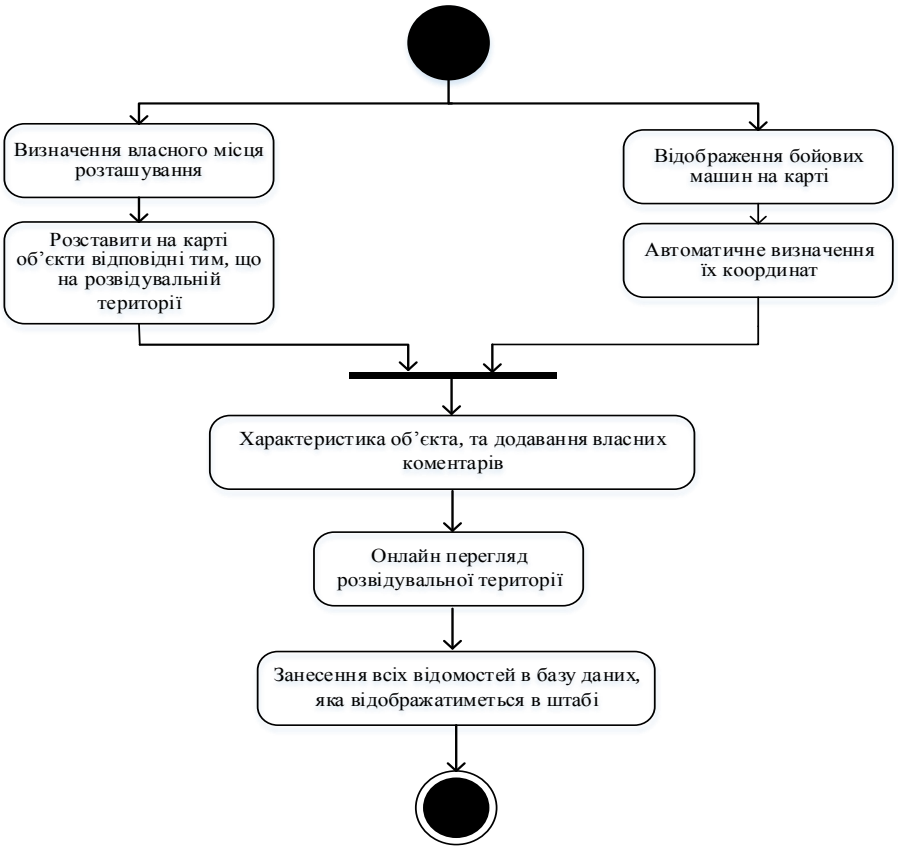

Рис. 3.22. Блок-схема застосунку «Millitary inteligence»

3.4.2. Імітаційне моделювання

Нехай для етапу орієнтація необхідно дослідити закономірності результату бою танкової роти (батальйону) при наступі на протитанковий укріплений район.

В ході виконання роботи було створено програму, яка взаємодіючи з онтологією здійснює прогноз результатів бою.

Основні дані для моделювання перебігу бойових дій зберігаються в онтології. Детальніше побудову таких систем описано в роботах [14-18, 24, 44, 137-149, 151]. На рис. 3.23 наведено UML діаграму класів для представлення поля бою, побудовану на основі даних онтології.

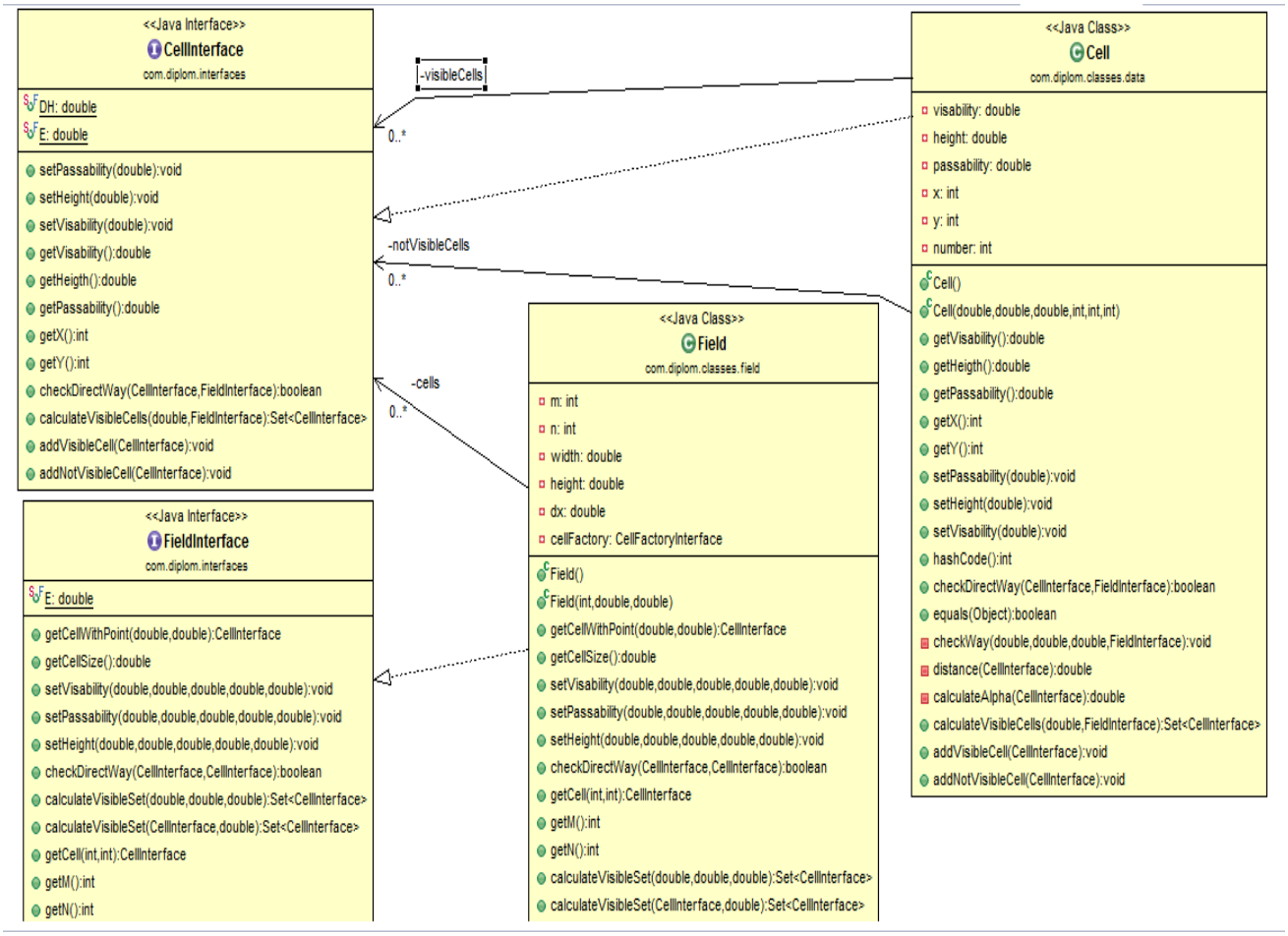

Рис. 3.23. Онтологічна модель представлення поля бою

Рис. 3.24. Блок-схема імітаційного моделювання

На рис. 3.24 побудовано алгоритм роботи даної системи.

3.4.3. Цілерозподіл на основі генетичних алгоритмів

Вдосконалюючи етап прийняття рішення ми використали задачу цілерозподілу. Цілерозподіл – це операція, яка полягає в призначенні певної цілі певному вогневому засобу [150, 151]. Необхідно знайти оптимальний (найкращий) цілерозподіл, призначивши кожному засобу ураження певну ціль, по якій вона повинна стріляти (при цьому можливо, що одна і та ж ціль буде обстріляна кількома засобами). Для розв’язування задачі пропонується використати методи штучного інтелекту, а саме генетичні алгоритми. Такі алгоритми використовуються для вирішення задач оптимізації і моделювання шляхом послідовного відбору, комбінування і варіації шуканих параметрів з використанням механізмів, які схожі на біологічну еволюцію (рис. 3.25).

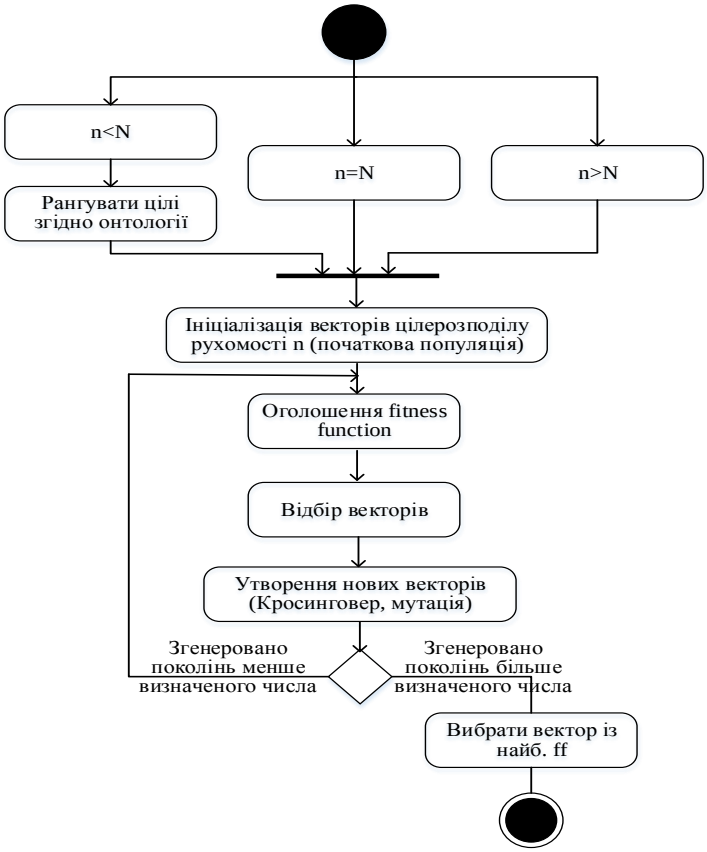

Рис. 3.25. Блок-схема генетичного алгоритму для задачі цілерозподілу

Вхідними даними для модуля цілерозподілу є ймовірнісні оцінки знищення противника в залежності від вогневого засобу, відстані до противника, різного роду коефіцієнтів (видимості, місцевості, прохідності тощо). Ці ймовірнісні величини зберігаються в онтології й визначені на основі нормативних документів. На виході модуля отримуємо вектор, елементами якого є пара («наш вогневий засіб (*M*)» – «ціль противника (*max*)»).

3.5. Архітектура СпППР

Підсистеми та модулі автоматизованої системи управління СВУ наведено на рис. 3.26. АСУ складається із таких модулів [104]:

1. Чотири сховища даних, три з яких поданих у вигляді онтології ПО та геоінформаційних систем. Так структура СВ України подається у вигляді дерева (графу), починаючи від найменшої структурної одиниці (взвод), закінчуючи армією. Техніка, озброєння, характеристики подається у вигляді

бази знань системи. Така база знань складається із бази даних ТТП різного роду озброєнь та правил виведення нових можливостей озброєнь. Сховище даних «Досвід» задається у виляді SWRL правил онтології СВУ.

2. Задачі. В сукупності перших два сховища даних дають змогу визначити перелік задач, які можуть виконати окремі підрозділи в залежності від технічного оснащення. Такий перелік задач подається у вигляді онтології задач.

3. Ситуація моделюється як підмножина наших військ та противника з прив'язкою до місцевості, яка задається у вигляді геоінформаційної системи. Прийняття рішень згідно до ситуації здійснюємо на основі підсистеми «Прийняття рішення», виходячи із мети наших військ (знищення сил противника, розвідка, дезінформація тощо).

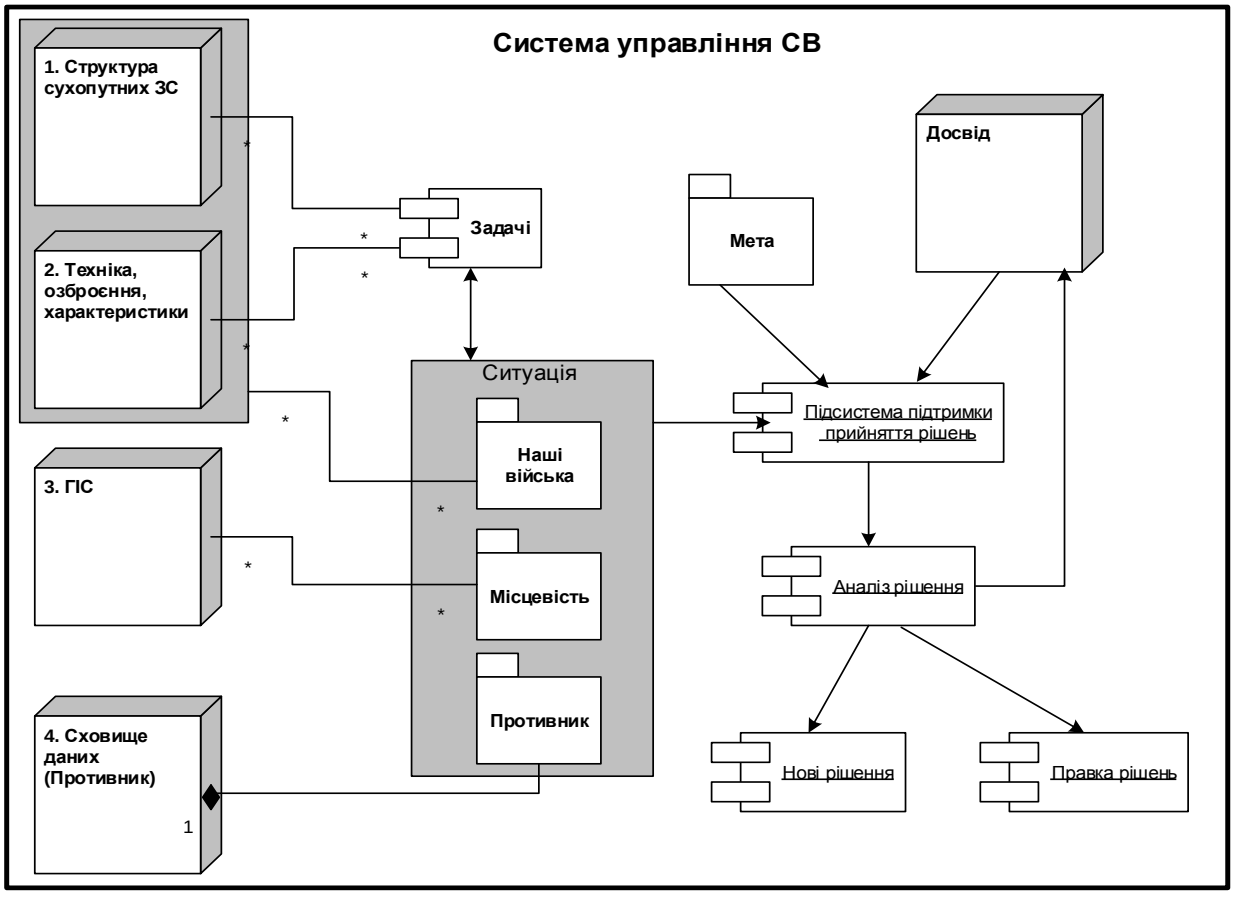

Рис. 3.26. Архітектура АСУ сухопутними військами

4. Прийняття рішення подається у вигляді петлі OODA, а саме так, як це наведено на рис. 3.26.

5. Аналіз рішення – експертне заключення прийнятих рішень. Використовується імітаційне моделювання різного роду ситуацій. Результатом такого моделювання є правка рішень, збереження попередніх рішень (досвід) або вироблення нових рішень.

Розроблена нами система підтримки прийняття рішень, яка реалізує петлю OODA у взаємодії з онтологією, включає такі модулі (рис. 3.27): розвідка та передача даних (мобільний застосунок «Military intelligence»); імітаційне моделювання бою; цілерозподіл з використанням генетичних алгоритмів; та на заключному етапі петлі OODA отримаємо коригування вогню, представлене у вигляді мобільного застосунку «Adjustment».

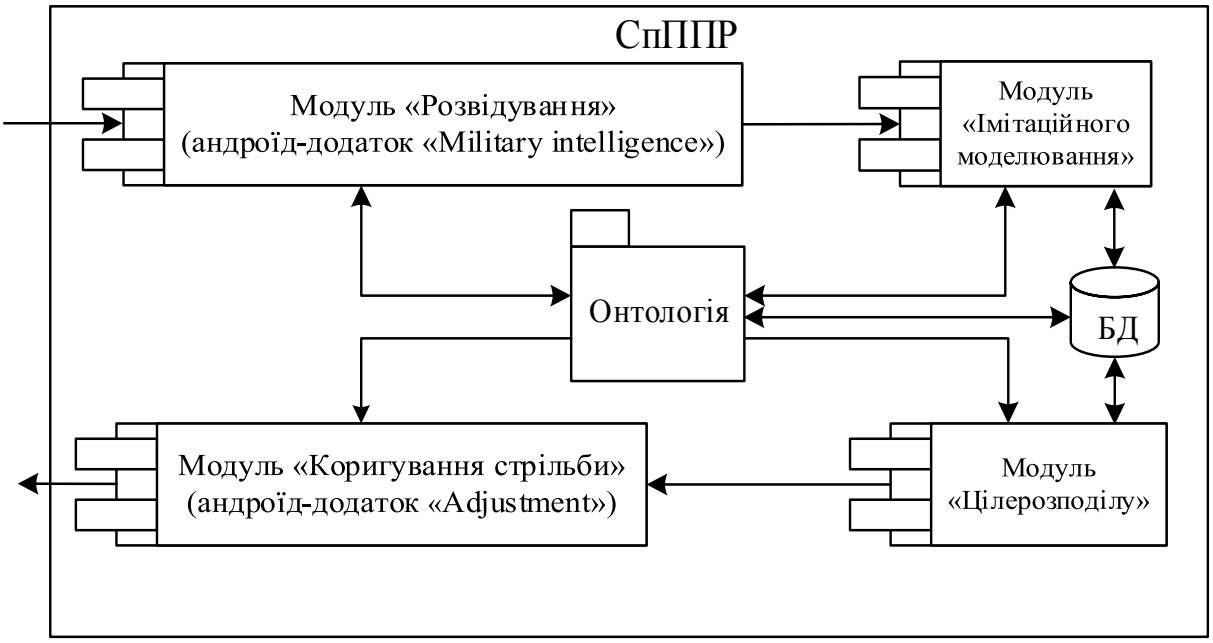

Рис. 3.27. Архітектура СпППР

3.6. Висновки до розділу 3

У цьому розділі відображено такі результати:

- розроблено онтологію СВ ЗСУ. Інформаційними джерелами побудови онтології є нормативні документи, які використовуються у Сухопутних військах, тактико-технічні характеристики бойових машин та озброєнь. Також використано експертні знання для побудови правил поведінки в

певних ситуаціях на основі описової логіки. Структура онтологічної моделі військових технологій включає чотири основних рівня: онтологію подання знань, онтологію технологій, онтологію військових технологій та прикладні онтології військових технологій. Основні концепти онтології військових технологій пояснюються моделлю збройної боротьби, побудованої на основі петлі Бойда;

- використано модель адаптивної онтології для задання важливості цілей противника. Таку модель використано під час побуди модулів імітаційного моделювання та цілерозподілу для ранжування цілей противника. В якості ваг важливості елементів онтології використано експертні оцінки;
- розроблено архітектуру СпППР та алгоритми функціонування програмних модулів СпППР на різних етапах петлі OODA, а саме: модуль розвідування, модуль імітаційного моделювання, модуль цілерозподілу на основі генетичного алгоритму, модуль корегування стрільби, а також БЗ, подана у вигляді онтології та БД, у якій зберігається поточний стан власних військ та необхідна довідникова інформація.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ СпППР У КОКУРЕНТНОМУ СЕРЕДОВИЩІ

У четвертому розділі наведено результати практичного впровадження запропонованих методів та засобів.

4.1. Опис розроблених модулів

У склад розробленої СпППР входять такі модулі: збору, обробки та передачі розвідувальних даних (мобільний застосунок «Military intelligence»), імітаційного моделювання перебігу бойових дій, цілерозподілу, корегування стрільби (мобільний застосунок «Adjustment»).

4.1.1. Мобільний застосунок «Military intelligence»

Розвідка та передача даних відбувається за допомогою мобільного застосунку «Military intelligence», що надає такі можливості:

- дає змогу швидко визначити власне місце розташування;
- розставити на карті об’єкти відповідні тим, що на розвідувальній території;
- автоматичне визначення їх координат;
- можливість охарактеризувати об’єкт, використовуючи піктограми військових стандартів та доданням власних коментарів;
- можливість надання штабу онлайн перегляду розвідувальної території;
- занесення всіх відомостей в базу даних, яка відображатиметься в штабі.

Вимоги до апаратно-програмного забезпечення:

- операційна система Android версії 4.0.4 та вище;
- разове підключення до мережі Інтернет;
- потрібно не менше ніж 10 Мб вільного місця в пам’яті пристрою.

Безпека передачі даних через мережу Інтернет відбувається за допомогою IPSec (Internet Protocol Security), комплекс, що включає в себе цілий набір протоколів: протокол безпеки зв’язку IKE; протоколи шифрування (DES, 3DES,

AES, RSA); протоколи перевірки достовірності даних (MD5, SHA-1); протокол обміну ключами (DH).

Після запуску програми на карту наноситься об’єкт, що спостерігається (рис. 4.1а), також можна охарактеризувати цей об’єкт та визначити його місце розташування (рис. 4.1б). Додаток дає змогу вивантажити дані, очистити карту, здійснити відео трансляцію обстежуваної території, зберегти дані в пам’яті пристрою, або надіслати на автоматизоване робоче місце (АРМ) командиру тактичної ланки.

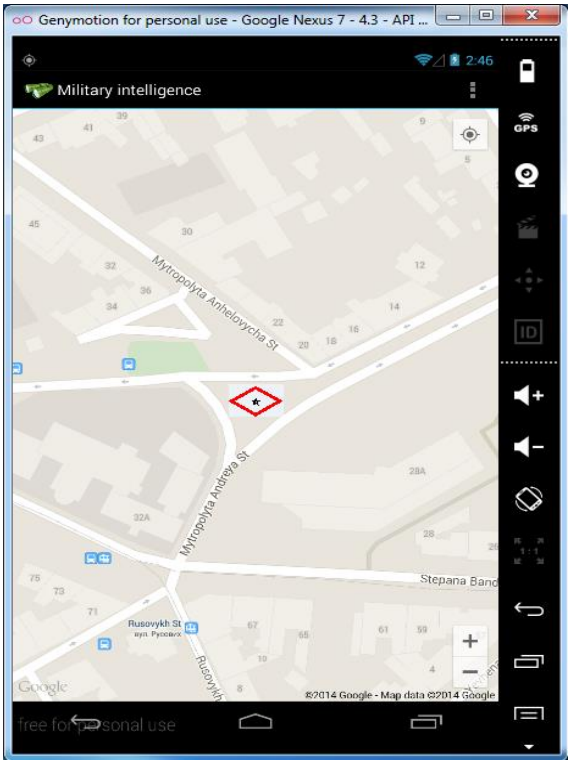

Рис. 4.1а. Розміщений об’єкт на карті

Рис. 4.1б. Характеристика об’єкту

В якості карт використано Google Maps. Для роботи з Google Maps були використані такі класи: GeoPoint – клас для роботи з координатами (вони представлені у вигляді довготи і широти); MapActivity – абстрактний клас який розширює діяльність; MapController – клас, службовець для масштабування і панорамування карти; MapView – в’юшки для відображення карти; OverlayItem – клас для роботи з маркером; TrackballGestureDetector – клас для роботи з MotionEvent; Overlay – базовий клас дозволяє працювати з маркерами; MapView.LayoutParams – в основному містить константи для відображення на

карті (по центру, з ліва і т.д.); ItimizedOverlay – абстрактний клас який містить список маркерів; MyLocationOverlay – клас для відображення вашого місцерозташування.

Таблиця 4.1. Категорії повідомлень, команд та сигналів відносно необхідної реакції командира, який їх отримує

№ з/п	Номер категорії (групи)	Опис категорії (групи)
1	2	3
1.	Категорія 1	повідомлення не відображається на електронній карті і в командному вікні АРМ при надходженні, але зберігається на АРМ і при відповідному запиті оператора АРМ відображається; не супроводжується звуковим сигналом; не вимагає негайної зворотньої реакції оператора АРМ, який отримав повідомлення.
2.	Категорія 2	повідомлення відображається на електронній карті (якщо має відповідні елементи пов’язані з нею) та в командному вікні; не супроводжується звуковим сигналом; не вимагає негайної зворотньої реакції оператора АРМ, який отримав повідомлення.
3.	Категорія 3	повідомлення відображається на електронній карті (якщо має відповідні елементи пов’язані з нею) та в командному вікні; супроводжується звуковим сигналом; не вимагає негайної зворотньої реакції оператора АРМ, який отримав повідомлення.
4.	Категорія 4	повідомлення відображається на електронній карті (якщо має відповідні елементи пов’язані з нею) та в командному вікні; супроводжується звуковим сигналом; при надходженні повідомлення на екрані АРМ відображається окреме вікно, яке тимчасово блокує попередню роботу оператора АРМ та вимагає підтвердження, що повідомлення прочитано.
5.	Категорія 5	повідомлення відображається на електронній карті (якщо має відповідні елементи пов’язані з нею) та в командному вікні; супроводжується звуковим сигналом; при надходженні повідомлення на екрані АРМ відображається окреме вікно, яке тимчасово блокує попередню роботу оператора АРМ та вимагає підтвердження, що повідомлення прочитано; додатково вимагається (або пропонується) зворотня реакція (рішення, відповідь) командира на отримане повідомлення.

Факт надходження повідомлень всіх категорій реєструється в журналі вхідних повідомлень (електронний журнал, який ведеться автоматично програмним забезпеченням, що встановлено на автоматизованому робочому місці) (табл. 4.1).

4.1.2. Модуль імітаційного моделювання

Надалі, згідно до отриманих даних про противника, командир моделює розташування власних військ. Для цього використовується модуль імітаційного моделювання перебігу бойових дій. На карту наносяться власні та чужі війська й запускається відповідний модуль.

Імітаційне моделювання розглядає наступне тактичне завдання. Танкова рота «синіх», що складається з m бойових машин, повинна прорвати протитанковий укріплений район «червоних». Цей район обороняє n бойових машин «червоних», які замасковані і знаходяться в спеціально створених укриттях. Рота повинна наступати в заданому бойовому порядку в смузі шириною a метрів і глибиною c метрів. Загальний напрямок руху роти визначається взаємним розташуванням на місцевості вихідної позиції «червоних» і «синіх». Напрямок руху для кожної бойової машини «синіх» визначається сукупністю орієнтирів, якими є добре розрізнювані місцеві предмети.

В ході виконання роботи було створено програму, яка взаємодіючи з онтологією здійснює прогноз результатів бою. Нижче наведено перелік, призначення та опис основних інтерфейсів програми (табл. 4.2 – табл. 4.8):

- CellInterface – призначений для представлення елементарної ділянки на полі бою. Класи, що реалізують цей інтерфейс інкапсулюють дані про видимість, прохідність, висоту над рівнем моря цієї ділянки;
- MashineInterface – призначений для опису бойової машини. Класи, що реалізують цей інтерфейс інкапсулюють дані про бойову машину(швидкострільність, тип засобу, потужність двигуна іт.д.), боєздатність, поточну позицію;

- FieldInterface – призначений для опису поля бою. Класи, що реалізують цей інтерфейс інкапсулюють дані елементарні ділянки та розміри поля бою;
- ModelInterface – описує тип моделі, яка використовується в СпППР;

WeaponInterface – описує зброю, яка може розміщуватися на бойовій машині; класи, що реалізують цей інтерфейс інкапсулюють дані про розсіювання по *x* і *y*, радіус стрільби та швидкострільність;

- WayInterfalInterface – описує інтервал шляху руху бойової машини;
- ShotInformationInterface – призначентй для опису інформації про результати пострілу;
- GranateInterface – призначений для опису снарядів; класи, що реалізують цей інтерфейс інкапсулюють дані про радіус ураження.

Таблиця 4.2. Опис методів інтерфейсу CellInterface

Назва методу	**Призначення**
void setPassability(double passability);	Встановити значення прохідності passability для елементарної ділянки.
voidsetHeight(double height);	Встановити значення висоти height для елементарної ділянки.
void setVisability(double visability)	Встановити значення висвидимості visabilit для елементарної ділянки.
double getVisability()	Отримати значення видимості даної ділянки.
double getHeigth()	Отримати значення висоти даної ділянки.
double getPassability()	Отримати значення прохідності даної ділянки.
int getX()	Отримати значення абсциси елементарної ділянки.
int getY()	Отримати значення оржинати елементарної ділянки.
boolean checkDirectWay(CellInterface cell1,FieldInterface field)	Перевірити чи існує прямий шлях до ділянки cell1 на полі field.
Set<CellInterface> calculateVisibleCells (double radius,FieldInterface field)	Повернути всі видимі елементарні ділянки у радіусі radius на полі field.

Таблиця 4.3. Опис методів інтерфейсу MashineInterface

Назва методу	**Призначення**
int getId();	Отримати значення унікального ідентифікатора бойової машини
void move(double t);	Знайти наві координати x і y машини в момент часу t.
double getX();	Отримати координату x машини.
double getY();	Отримати координату y машини.
void setX(double x);	Встановити x координату машини.
void setY(double y);	Встановити y координату машини.
void addWayInterval(double x,double y);	Додати інтервал шляху з кінцевою точкою (x, y).
void removeWay();	Видалити шлях, яким має їхати бойова машина
void removeLastWayInterval();	Видалити останній інтервал шляху.
void watch(FieldInterface field,List<MachineInterface > mashines, double t);	Перейти в режим спостерігача і виявляти машини mashines на полі field протягом t секунд.
MachineInterface shot(FieldInterface field, double t);	Вибрати ціль і здійснити на полі field в момент часу t. Якщо постріл здійснено і ціль уражена, то повернути посилання на об’єкт ураженої цілі, інакше повернути null.
boolean isButtleWorthy();	Отримати значення боєздатності.
void kill();	Зробити машину небоєздатною.
boolean getColor();	Отримати колір машини(одна команда має значення кольорів машин true інша false).
double getVisabilityRadius();	Отримати радіус огляду.
void paveWay (int time,FieldInterface field);	Прокласти шлях машини на наступні t секунд на полі field.
void reset();	Обнулити всі дані про бойову машину(зробити боєздатною, встановити початкові координати) для наступної ітерації моделювання.
String getName();	Отримати ім’я бойової машини.
double getShotRadius();	Отримати радіус стрільби бойової машини.

Таблиця 4.4. Опис методів інтерфейсу FieldInterface

Назва методу	**Призначення**
CellInterface getCellFromPoint (double y, double x);	Отримати елементарну ділянку, якій належить точка з координатами (x, y).
double getCellSize();	Отримати розмір елементарних ділянок.
void setVisability (double x,double y,double width,double height,double value);	Для всіх комірок, які попадають в прямокутник з координатами лівого кута (x, y), та значеннями ширини і висоти width та height встановити значення видимості рівним value.
void setPassability (double x,double y,double width,double height,double value);	Для всіх комірок, які попадають в прямокутник з координатами лівого кута (x, y), та значеннями ширини і висоти width та height встановити значення прохідності рівним value.
void setHeight (double x,double y,double width,double height,double value);	Для всіх комірок, які попадають в прямокутник з координатами лівого кута (x, y), та значеннями ширини і висоти width та height встановити значення висоти рівним value.
boolean checkDirectWay(CellInterface cell1,CellInterface cell2);	Перевірити чи існує прямий шлях між ділянками cell1 та cell2.
Set<CellInterface> calculateVisibleSet(double y,double x,double radius);	Отримати множину елементарних ділянок, які видимі з точки з координатами (x, y) у радіусі radius.
CellInterface getCell (int y,int x);	Отримати елементарну ділянку з координатами (x, y).
int getM();	Отримати кількість комірок по висоті поля.
int getN();	Отримати кількість комірок по ширині поля.

Таблиця 4.5. Опис методів інтерфейсу ModelInterface

Назва методу	**Призначення**
ModelResultInterface model();	Здійснити моделювання бою та повернути результати моделювання.
void addTank (MachineInterface tank);	Додати до моделі бойову машину tank.
void setField (FieldInterface field);	Використовувати в якості поля бою поле field.
BestCellsInterface calculateBesyCells();	Обчислити і повернути найкращу(ту з якої можна уразити найбільше бойових машин і отримати найменше шкоди) позицію.

Таблиця 4.6. Опис методів інтерфейсу WeaponInterface

Назва методу	Призначення
ShotInformationInterface shot(GranateInterface granate,double distance,double recomendedX, double recomenendetY);	Здійснити постріл (recomendedX, recomenendetY), яка знаходиться на відстані distance, снарядом granate та повернути інформацію(реальні координати куди попав снаряд) про постріл.
double getRate();	Отримати значання швидкострільності.
double getNextShotTime()	Отримати час наступного пострілу.
double getShotRadius();	Отримати радіус стрільби.
void reset();	Скинути час наступного пострілу в 0.

Таблиця 4.7. Опис методів інтерфейсу ShotInformationInterface

Назва методу	Призначення
double getX();	Отримати значення абсциси точки попадання снаряду.
double getY();	Отримати значення ординати точки попадання снаряду.

Таблиця 4.8. Опис методів інтерфейсу GranateInterface

Назва методу	Призначення
Double getRadius();	Отримати значення радіусу ураження снаряду.

Клас Cell реалізує інтерфейс CellInterface, а клас Field – FieldInterface. Клас Field містить у полі cells 0 або багато об’єктів, що реалізують CellInterface. Клас Cell містить у полях visibleCells і notVisibleCells 0 або багато об’єктів, що реалізують CellInterface.

Класи та інтерфейси, які призначені для представлення бойових машин клас Mashine реалізує інтерфейс MashineInterface, клас Granate – GranateInterface, клас Weapon реалізує WeaponInterface, а WayInterval реалізує WayIntervalInterface. Клас Mashine містить у полі ways 0 або багато об’єктів, що реалізують WayIntervalInterface, 0 або або багато об’єктів, що реалізують MashineInterface у полі watchMashines, 0 або 1 об’єкт, що реалізує GranateInterface та WeaponInterface у полях granate і weapon відповідно.

Клас StohasticModel реалізує інтерфейс ModelInterface, клас ModelResult – ModelResultInterface, клас BestCells реалізує BestCells Interface, а BestCellFinder реалізує BestCellFinder lInterface. Клас StohasticModel містить у полі bestCellFinder 0, або 1 об’єкт, що реалізують BsetCellFinderInterface.

Основні дані для моделювання перебігу бойових дій зберігаються в онтології. Детальніше побудову таких систем описано в роботах [14–18].

Бій починається в деякий заданий час T_0 і триває до того моменту, коли сили однієї з сторін, що борються стануть небоєздатними. Кінцем бою можна також вважати той момент $T_1 > T_0$, в який буде виконана бойове завдання або втрати однієї зі сторін перевищать допустимий відносний рівень. Головне вікно програми з картою та взаємним розміщенням бойових машин зображено на рис. 4.2.

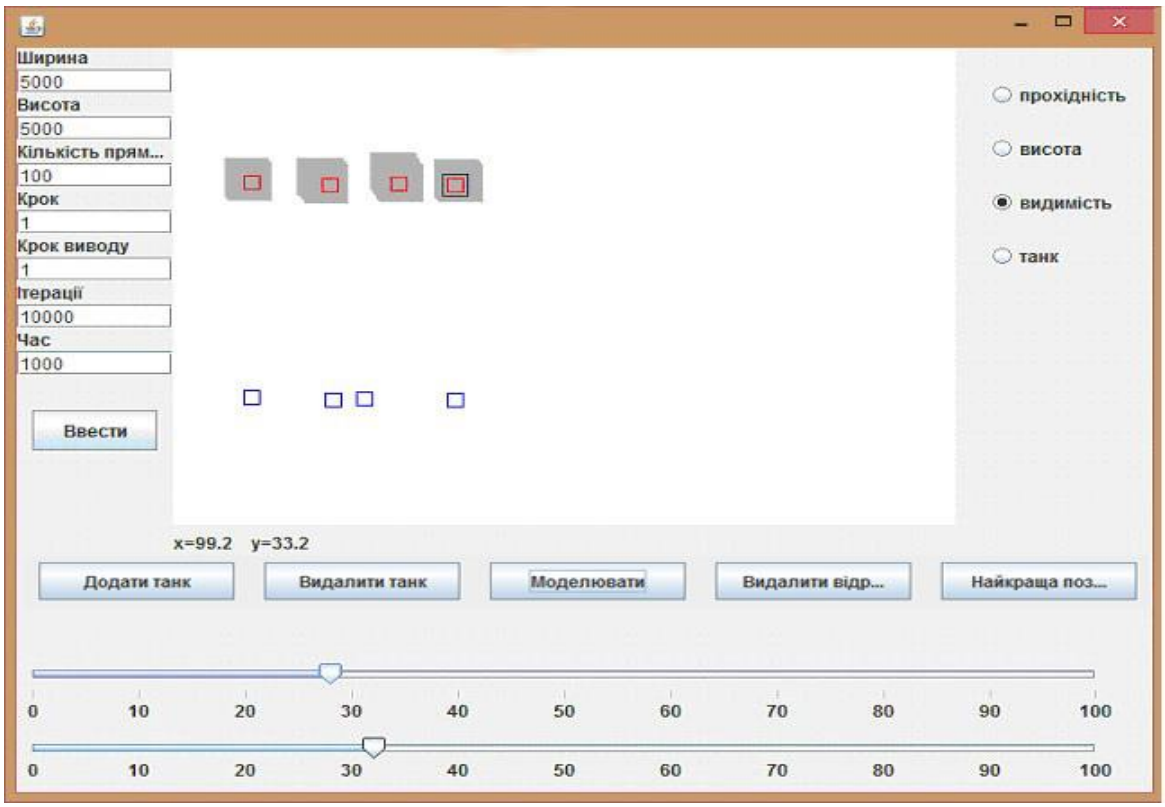

Рис. 4.2 Головне вікно програми

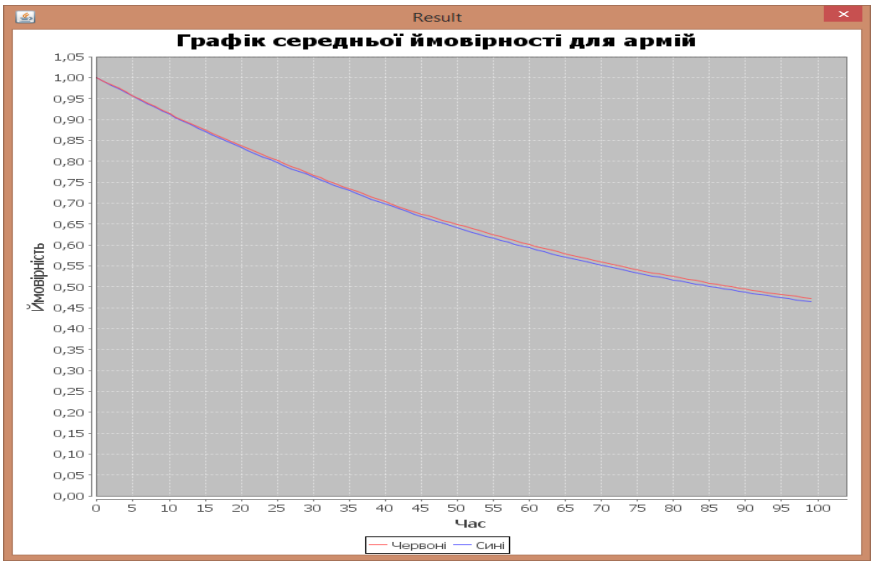

Рис. 4.3. Графік середньої ймовірності

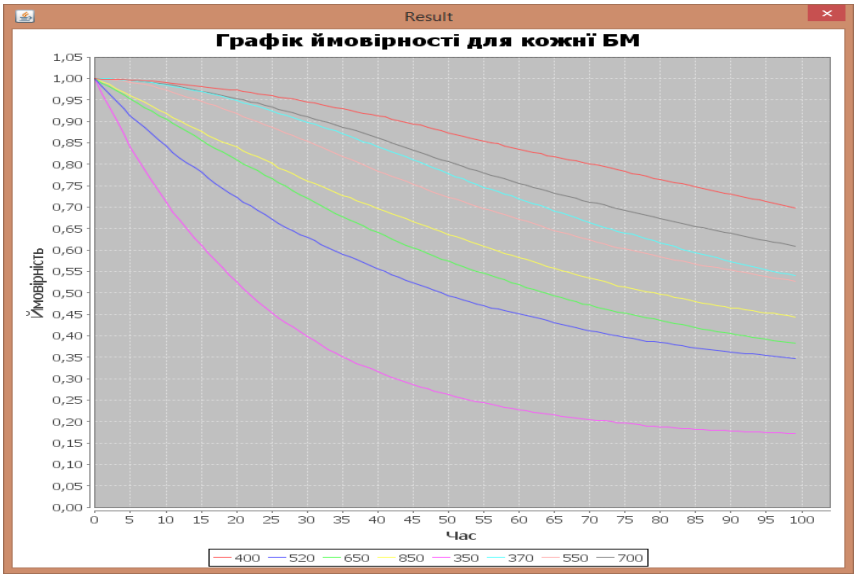

Рис. 4.4. Графік ймовірності для кожної БМ

З графіків наведених на рис. 4.3 та рис. 4.4 видно, що бій розпочнеться приблизно в момент T_0. Бій закінчується в момент T_0+100c. Починаючи з цього моменту графік ймовірностей стає паралельним, до осі x, отже, це кінець бою. Обчислимо ймовірності виграшу кожної із сторін:

$$P\ (\text{виграш} = \text{«сині»}) = \frac{0,2}{0,4+0,2} = 0,33.$$

$$P\ (\text{виграш} = \text{«червоні»}) = \frac{0,2}{0,4+0,2} = 0,66.$$

Колір лінії відповідає певній бойовій машині танкової роти «синіх» чи «червоних». Більшу ймовірність виграшу «червоних» можна пояснити кращою захищеністю їх військ, оскільки за умовою задачі «червоні» замасковані і знаходяться у спеціально створених укриттях. Основні дані для моделювання перебігу бойових дій зберігаються в онтології. Модуль також дає змогу знаходити найкращу позицію розташування БМ під час бою. В якості карт місцевості використано геоінформаційну систему «ArcGIS». Для окремих ситуацій, використання експертних правил, які містяться в онтології, дає змогу до 20 % підвищити ймовірність неушкодження БМ.

Розмістимо тепер на карті дві однакові бойові машини («червону» і «синю»), так щоб відстань між ними не перевищувала радіус пострілу і позначимо добре замасковану ділянку. Цю ділянку виберемо так, щоб жодна бойова машина не знаходилася в ній (рис. 4.5).

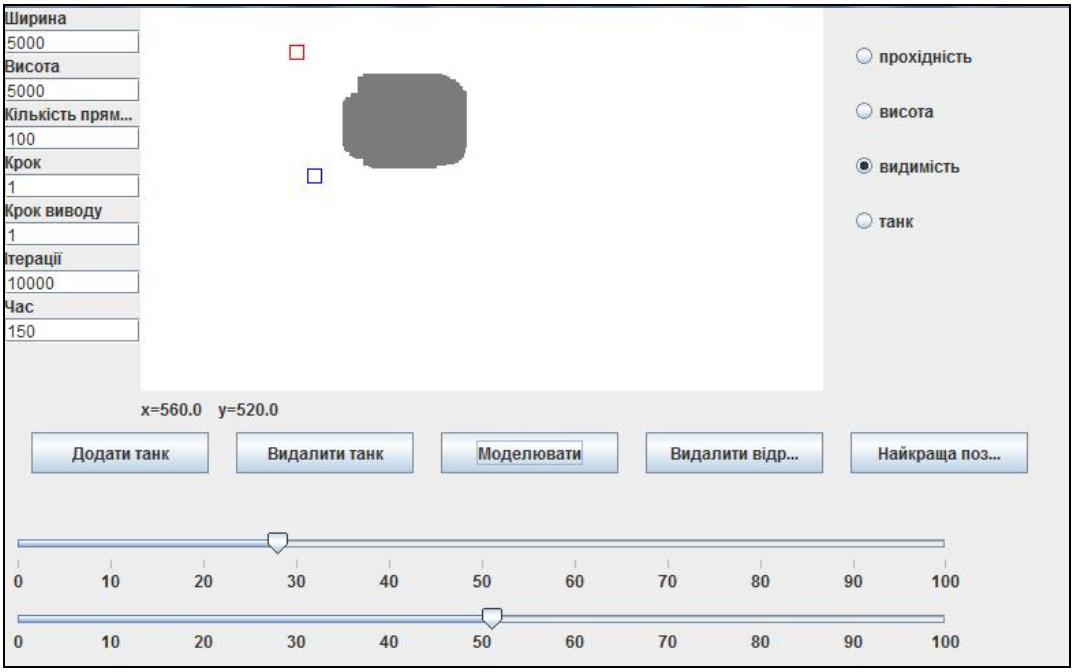

Рис. 4.5. Дуельний бій

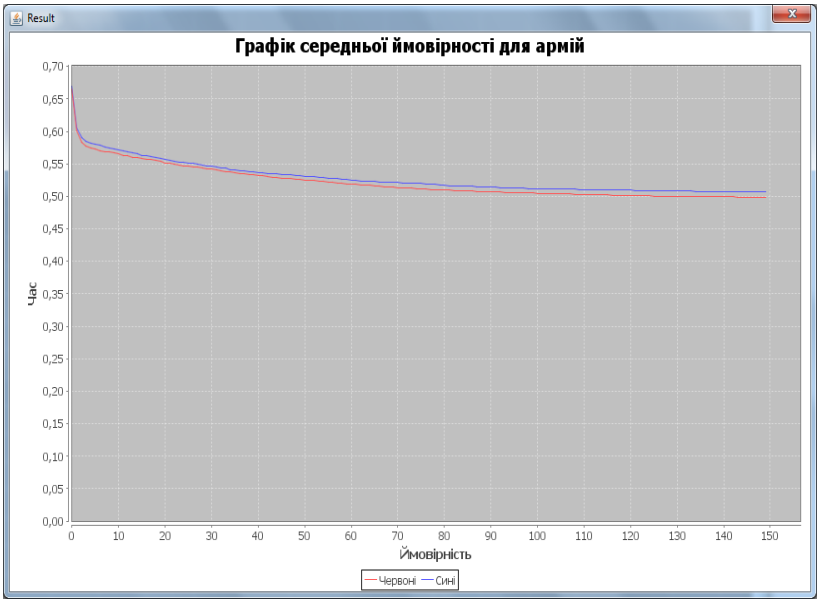

Рис. 4.6. Результати дуельного бою

На рис. 4.6 бачимо динаміку бою. Оскільки бойові машини однакові, то ймовірності їх виграшу теж однакові. Незначну перевагу синіх можна пояснити похибкою моделювання.

Знайдемо найкращу позицію для атаки (рис. 4.7). Найкраща позиція зображена заштрихованою ділянкою.

Тепер переставимо «синю» бойову машину в цю позицію і знову спрогнозуємо результати. З наведеного графіка видно, що ймовірність виграшу синіх значно зросла (рис. 4.8).

З наведених прикладів можна зробити висновок, що програма працює правильно у різноманітних ситуаціях.

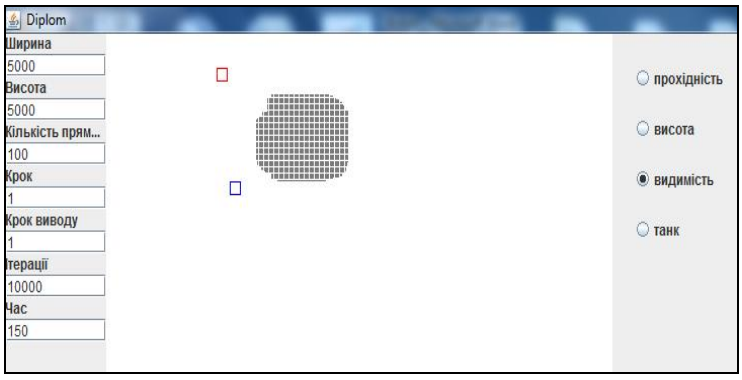

Рис. 4.7. Найкраща позиція

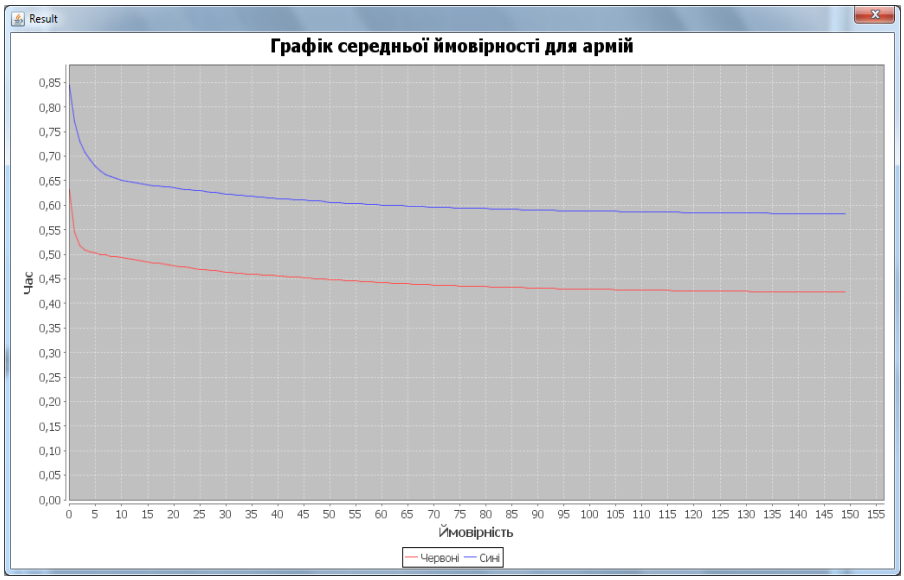

Рис. 4.8. Прогноз динаміки бою

4.1.3. Модуль цілерозподілу

Вхідними даними для модуля цілерозподілу є ймовірнісні оцінки знищення противника в залежності від вогневого засобу, відстані до противника , різного роду коефіцієнтів (видимості, місцевості, прохідності тощо). Ці ймовірнісні величини зберігаються в онтології й визначені на основі нормативних документів. На виході модуля отримуємо вектор, елементами якого є пара («наш вогневий засіб» – «ціль противника»). Для розв’язку цієї задачі ми використали генетичний алгоритм [130, 131].

Для реалізації запропонованого підходу обрано БД реляційного типу (а саме MySQL), в якій зберігається інформація про наші наявні засоби ураження, розвідані цілі противника, та матриця ймовірностей знищення цілей певним вогневим засобом. Хромосома являє вектор, де номер елемента вектора – відповідає ключу засобу знищення в базі даних, а значення елемента – ключ цілі в базі даних.

Кращим розв’язком вважається хромосома з найбільшим значенням функції пристосованості.

Для моделювання обиралась певна кількість генерації поколінь хромосом. Результати експериментів показали, що при генерації 30 поколінь

знайдена найкраща хромосома близька до оптимального цілерозподілу. Розроблений модуль цілерозподілу на основі генетичних алгоритмів входить в склад СпППР.

Основною перевагою запропонованого підходу є значне зменшення складності алгоритму цілерозподілу. Складність повного перебору – експотенційна, а генетичного алгоритму – лінійна. Тим самим набагато прискорюється процес прийняття рішень. Особливо це важливо, коли події щодо цілерозподілу відбуваються в реальному часі. Хоча отриманий розв'язок цілерозподілу не є завжди оптимальним, однак він близький до оптимального, а виграш в часі отримання розв'язку є значним.

Модуль реалізовано мовою програмування Java. Текст модуля наведено у додатку.

4.1.4. Модуль коригування стрільби

Коригування стрільби відбувається за допомогою мобільного застосунку «Adjustment», що дає змогу здійснити пристрілку цілі. Розрахунки вирахуваних установок для ураження цілі можна здійснювати окомірно, скорочено, повною підготовкою. Однак кожний з таких методів має свої недоліки, а саме мала точність, або значні витрати часу. Під час пристрілки враховуються метео- і балістичні відхилення для топографічних даних до цілі: $D_v = D_t + \Delta D$, де D_v – дальність вирахувана, D_t – дальність топографічна, ΔD – сумарна поправка дальності на метео-, балістичні відхилення. Приціл, що відповідає D_v, забезпечує проходження траєкторії снаряда через центр цілі. У розробленому застосунку використано пристрілку за Графіком. Таку пристрілку доцільно використовувати у випадках, коли ціль знаходиться на вершині, або крутому схилі місцевості. Суть пристрілювання за Графіком, полягає в тому, що трьома пострілами на різних установках прицілу й напрямку в районі цілі позначається площина стрільби і створюється масштаб для визначення відхилень за дальністю і напрямком у метрах.

Сітку визначення коректур для пристрілювання за допомогою мобільного застосунку «Adjustment» будують на Canvas. Для побудови сітки проводять дві взаємноперпендикулярні лінії, що для корегувальника відповідають осям: лінії спостереження X (дальність) і напрямку Y (бокові відхилення).

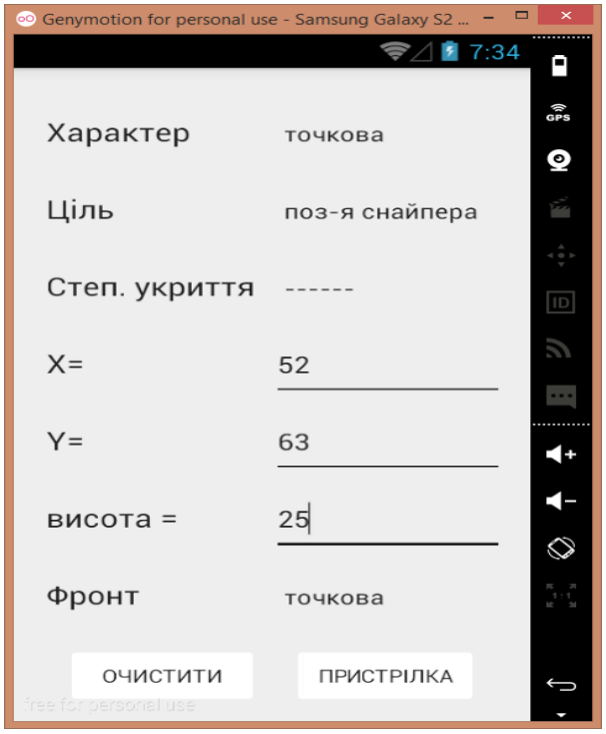

Рис. 4.9а. Характеристика цілі та координати позиції снайпера

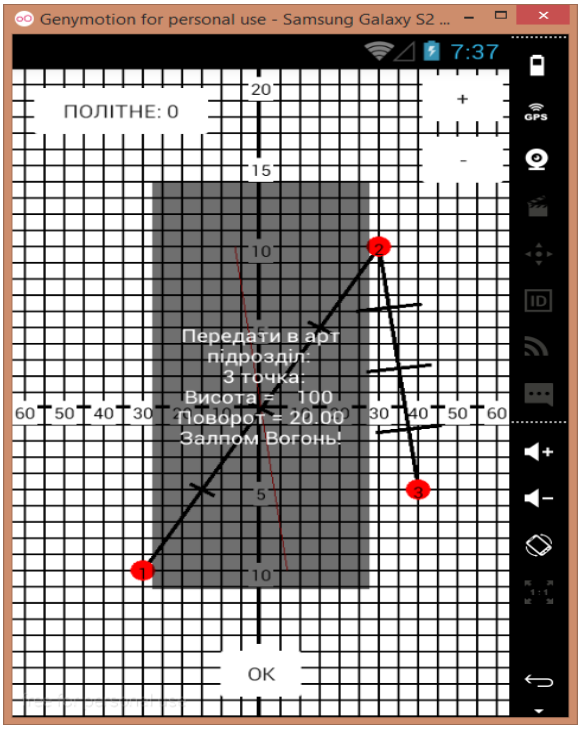

Рис. 4.9б. Поправка до цілі

Мобільний застосунок «Adjustment» надає такі можливості: дає змогу військовослужбовцю (не артилеристу) передати в артилерійський підрозділ інформацію про розвідану ціль (рис. 4.9а); у ході пристрілки коригувальник, подає команди згідно вимог керівних документів артилерії; військовослужбовцем будь-якого загальновійськового підрозділу здійснити пристрілку цілі та забезпечити визначення установок для стрільби на ураження цілі (рис. 4.9б.); використовуючи масштаби дальності та напрямку, здійснити обстріл нової цілі, яка знаходиться у радіусі до 1 км від пристріляної; не потрібно визначати та передавати координати коригувальника, що усуває можливість противнику визначити його місце знаходження; масштабувати зображення на дисплеї, а також виводити таймер зворотного відліку часу польоту снаряду (польотний час може бути до 65 секунд, тому тривале

спостереження в оптичний прилад значно стомлює зір коригувальника). Вимоги до апаратно-програмного забезпечення та безпека передачі даних аналогічні, як у застосунку «Military intelligence».

Виконуючи вогневе завдання дивізіоном, пристрілювання цілі однією, або кожною батареєю ведуть згідно із зазначеним раніше порядком. Під час пристрілювання кожною батареєю залпи призначають із темпом, який забезпечує спостереження їх штурманом-коректувальником. Суть пристрілювання за шкалою полягає в тому, що двома групами розривів на різних установках прицілу в районі цілі позначається площина стрільби і створюється масштаб для визначення відхилень за дальністю і напрямком у метрах.

В програмі загалом використовуються 2 Activity: InputActivity (для вводу інформації про ціль) та GridActivity(для введення та відображення результатів пострілів). Промальовка здійснюється на CanvasView, використовуючи DrawManager.

Усі діалоги здійснені використовуючи dialogFragments (знаходяться в відповідному package) та викликаються через DialogManager.

Виділені наступні моделі:

- HitPoint (точка розриву снаряду) – містить інформацію про розміщення (x,y), номер пострілу, найближчий попередній постріл, кількість відрізків між точками, та тип координат які використовує точка (Андроїд чи Декартова).
- LinearLine (лінія між 2-ма точками) – містить початкову, кінцеву точки, кількість відрізків між точками, формулу утвореної прямої, значення прицілу та довороту, що були вказані на для пострілу.
- CanvasModel (інформація яка буде використовуватись для відображення на схемі) – містить розмір дисплею, поточну кількість відрізків між точками, набір ліній, набір точок, набір точок для паралельних ліній.
- Менеджери:

- DialogManager – для виклику усіх діалогів (від діалогу введення часу польоту до діалогу команди).
- CountDownManager – для роботи з таймером зворотнього відліку.
- DrawManager – для промальовки результатів пострілів. Є одним з найбільш важливих компонентів, оскільки вся промальовка на CanvasView здійснюється через його методи.
- Утиліти:
- CommandCreator – використовується для формування команд на основі вхідних параметрів.
- CoordinateSystem – відповідає за перетворення між координатними системами з Декартової до Андроїд та навпаки.
- MathUtil – як і DrawManager є однією з найважливіших у даному додатку. Містить у собі усі методи для розрахунку координат, ліній, найближчих точок і т.д. Будь які зміни у MathUtil потребують повного тестування додатку.

Базовий алгоритм розрахунку поправки, який використовується в програмі.

Поправка по Дальності:

Крок 1. Візьмемо Лінію #1 та величину Дальності Лінії #1 (200 або 400 залежно від введених даних) і паралельну лінію до Лінії #2, що проходить через центр координат (назвемо її Паралельна Лінія #2).

Крок 2. З точок Лінії #1 знаходимо найближчу до центра координат точку (назвемо її Точка А). Від цієї точки буде обраховуватись відстань (поправка по Дальності) до перетину з Паралельною Лінією #2.

Крок 3. Знаходимо точку перетину Лінії#1 та Паралельної Лінії #2 (назвемо її Точка Перетину #1).

Крок 4. Обраховуємо відстань між точками Лінії #1 (назвемо її Відстань між точками Лінії #1).

Крок 5. Обраховуємо відстань від Точки А до Точки Перетину #1 (назвемо її Відстань між точками обрахунку #1).

Крок 6. Обраховуємо поправку по Дальності за формулою:

Поправка по Дальності = Відстань між точками обрахунку #1 * Дальність Лінії #1 / Відстань між точками Лінії #1.

Поправка по Довороту:

Крок 1. Візьмемо Лінію #2 та величину Довороту Лінії #2 (20 або 40 залежно від введених даних) і паралельну лінію до Лінії #1, що проходить через центр координат (назвемо її Паралельна Лінія #1).

Крок 2. З точок Лінії #2 знаходимо найближчу до центра координат точку (назвемо її Точка В). Від цієї точки буде обраховуватись відстань (поправка по Довороту) до перетину з Паралельною Лінією #1.

Крок 3. Знаходимо точку перетину Лінії #2 та Паралельної Лінії #1 (назвемо її Точка Перетину #2).

Крок 4. Обраховуємо відстань між точками Лінії #2 (назвемо її Відстань між точками Лінії #2).

Крок 5. Обраховуємо відстань від Точки В до Точки Перетину #2 (назвемо її Відстань між точками обрахунку #2).

Крок 6. Обраховуємо поправку по Довороту за формулою:

Поправка по Довороту = Відстань між точками обрахунку #2 * Доворот Лінії #2 / Відстань між точками Лінії #2.

4.2. Аналіз отриманих результатів

У ході виконання дисертаційного дослідження, за допомогою розроблених методів і програмних засобів, експериментально доведено, ефективність розробленої СпППР, що дозволила до 30 % скоротити час, який витрачають командири тактичних ланок на на планування і доведення завдань до підлеглих; покращення тактичної і бойової підготовки військових кадрів СВ ЗСУ. Дані апробації наведені у табл.4.9.

Таблиця 4.9. Оцінка ефективності СППР

№ з/п.	Заходи щодо підготовки до оборони	Час без СпППР	Час з СпППР
1.	Усвідомлення завдання, проведення розрахунку часу, визначення заходів, які необхідно провести негайно. Віддання вказівок щодо підготовки до маршу та району бойового призначення, організація розвідки.	10	10
2.	Оцінка обстановки та прийняття рішення (**застосунок «Military intelligence»**, **імітаційне моделювання**)	15	5
3.	Оцінка обстановки (**застосунок «Military intelligence»**): • Ком. РВ – відомості про противника • НШ – відомості про свої підрозділи • Ст. Інж. – відомості про стан техніки та озброєння • ЗКМТЗ – відомості про запаси МТЗ • Командир ісв – відомості про місцевість	20	10
	Визначення замислу	15	15
	Віддання попередніх бойових розпоряджень	10	10
	Визначення бойових завдань підрозділам (**імітаційне моделювання**)	20	5
	Визначення основних питань взаємодії та організації управління	15	15
4.	Розроблення бойового наказу підрозділам та організація рекогносцировки	10	10
5.	Уточнення рішення на місцевості, участь у рекогносцировці, яку проводить командир, доповідь рішення на оборону (**цілерозподіл**)	20	10
6.	Віддача бойового наказу. Затвердження рішень командирів підрозділів.	20	20
7.	Організація взаємодії та управління (**цілерозподіл, застосунок «Adjustment»)**	15	10
	Всього	**3 год.**	**2 год.**

4.3. Висновки до розділу 4

Розроблено програмне забезпечення для кожного з етапів петлі OODA на основі побудованих методів та онтологічному підході. Для етапу «Спостереження» розроблено мобільний застосунок «Military intelligence», який опрацьовує розвідувальну інформацію й передає в штаб. Для реалізації етапу «Орієнтація» розроблено модуль імітаційного моделювання, вхідними даними

якого є отримана розвідувальна інформація та наявний стан власних військ. Для реалізації етапу рішення розроблено модуль ефективного цілерозподілу, вхідними даними якого є розвідувальні дані й онтологія військ. Для етапу «Дія» розроблено мобільний застосунок «Adjustment» корегування стрільби. Використання розроблених моделей, методів та програмних модулів дала змогу до 30 % скоротити час, який витрачають командири тактичних ланок на планування військових дій.

ВИСНОВКИ

У дисертаційній роботі вирішено важливе науково-прикладне завдання розроблення методів та засобів побудови системи підтримки прийняття рішень у конкурентному середовищі (військова сфера) з використанням онтологічного підходу та підвищення ефективності таких систем, якого досягнуто завдяки застосуванню розробленого математичного та програмного забезпечення, що ґрунтується на використанні онтологій у цих системах, адаптацією онтологій до специфіки задач предметної області. Під час виконання роботи одержано такі наукові та практичні результати.

1. Обґрунтовано доцільність розроблення математичних моделей, методів та засобів підтримки прийняття рішень у конкурентному середовищі на основі петлі Бойда з використанням онтологічного підходу в тих предметних областях, в яких знання є експліцитними. Такою предметною областю виступає військова сфера.

2. Розроблено модель петлі Бойда на основі автомата Мура. Станами автомата виступають етапи петлі Бойда, а також процеси наповнення, редагування онтології та пошук релевантних знань в онтології. Визначено можливі переходи між станами автомата і передача параметрів між ними. Розроблений автомат служить основою для побудови СпППР для командирів тактичних ланок СВ ЗСУ.

3. Для моделювання процесу підтримки прийняття рішень у конкурентному середовищі розроблено математичне забезпечення та методи використання онтології предметної області на чотирьох етапах петлі OODA (спостереження, орієнтація, рішення, дія). Так для військової сфери на етапі «Спостереження» розвідувальні дані аналізуються онтологією предметної області з метою визначення сильних та слабких сторін противника. На етапі «Орієнтації» онтологічні дані використовуються для імітаційного моделювання можливого перебігу бою й для оптимального розставлення власних сил. Для

етапу «Рішення» розроблено метод цілерозподілу на основі генетичних алгоритмів, що дало змогу зменшити обчислювальну складність пошуку ефективного цілерозподілу, тим самим пришвидшити час необхідний командиру тактичної ланки для прийняття рішення. Ймовірнісні дані знищення цілей противника певним засобом, беруться з онтології на основі аналізу вхідних параметрів (видимість, прохідність, швидкість, боєздатність, погодні умови). Також для підвищення ефективності можливих рішень в онтології подано експертні знання на основі дескриптивної логіки.

4. Розвинуто метод використання онтологій у прикладних предметних областях, а саме у військових застосуваннях, за рахунок використання дескриптивної логіки та визначення експертами ваг окремих елементів онтології, що дало змогу підвищити ефективність етапів «Орієнтація» та «Рішення» петлі Бойда під час імітаційного моделювання перебігу бойових дій та цілерозподілу. Для окремих випадків згідно з результатами імітаційного моделювання, використання експертних правил та оцінок елементів онтології, дає змогу до 20 % підвищити ймовірність неушкодження власних військ.

5. Розроблено архітектуру підсистеми підтримки прийняття рішень, яка складається із модулів, що задають відповідний етап петлі OODA. Центральною компонентою СпППР є онтологія предметної області. Побудовано таку онтологію для СВ ЗСУ. Розроблена онтологія містить 384 понять, 207 відношень. Здійснено визначення окремих елементів онтології засобами дескриптивної логіки, 27 % понять є визначеними. Екземпляри окремих понять онтології зберігаються в БД.

6. Розроблено програмне забезпечення для кожного з етапів петлі OODA на основі побудованих методів та онтологічного підходу. Для етапу «Спостереження» розроблено мобільний застосунок «Military intelligence», який опрацьовує розвідувальну інформацію і передає її в штаб. Для реалізації етапу «Орієнтація» розроблено модуль імітаційного моделювання, вхідними даними якого є отримана розвідувальна інформація, а на виході план розташування

власних військ. Для реалізації етапу «Рішення» розроблено модуль пошуку ефективного цілерозподілу, вхідними даними якого є ймовірності ураження цілей певним вогневим засобом, а на виході призначення вогневому засобу цілі противника, яку він обстрілює. Для етапу «Дія» розроблено мобільний застосунок «Adjustment» для корегування стрільби. Використання розроблених моделей, методів та програмних модулів дало змогу до 30 % скоротити час, який витрачають командири тактичних ланок на планування військових дій.

ЛІТЕРАТУРА

1. Angerman W. S. Coming full circle with Boyd`s OODA loop ideas: an analysis of innovation diffusion and evolution // Air Force Institute of Technology. – 2004.
2. Banzhaf W. Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications / W. Banzhaf, P. Nordin, R. Keller, F. Francone // Morgan Kaufmann. – 1997.
3. Bertino E. Intelligent Database Systems / E. Bertino, B. Catania, G. Zarri. – Addison-Wesley, 2001. – 452 p.
4. Biao Qin, Shan Wang, Xiaoyong Du, Qiming Chen, Qiuyue Wang Graph-based Query Rewriting for Knowledge Sharing between Peer Ontologies // Information Sciences, 2008. – 178(18). – P. 3525-3542.
5. Boyd J. R.. Organic design, Slideshow. URL: http://d-ni.net/boyd/strategic_game.pdf [Accessed 7th September, 2006].
6. Boyd J. R. Strategic game of ? and ?, Slideshow. URL: http://d-n-i.net/boyd/strategic_game.pdf [Online; accessed 7th September, 2006].
7. Bulskov H. On Querying Ontologies and Databases / H. Bulskov, R. Knappe, R.Andreasen // FQAS. – 2004. – P. 191-202.
8. Carsten Lutz. Description Logics with Concrete Domains – A Survey. In P. Balbiani, N.-Y.Suzuki, F.Wolter, and M.Zakharyaschev, editors, Advances in Modal Logics, Volume 4. King’s College Publications, 2003.
9. Description Logic Framework for Information Integration. In Proc. Of KR’98, P. 2-13, Morgan Kaufmann, San Francisco, 1998.
10. Diego Calvane, Guiseppe De Giacomo, Maurizio Lenzerini, Damiele Nardi, and Riccardo Rosati. DoD modelling and simulation (M&S) Glossary, march, 2010. URL: http://www.msco.mil/files/Draft_MS_Glossary_March_B_versio.pdf

11. Donini, Daniele Nardi and Riccardo Rosati. Description Logics of Minimal Knowledge and Negation as Failure // ACM Transactions on Computational Logic, 2002. – 3(2). – P. 177-225.
12. Dosyn D. G. Modelling of the intelligent text recognition agents based on dynamic ontology. / D. G. Dosyn, R. R. Darevych, V. V. Lytvyn // Proceedings of the 4th International Conference „Internet – Education – Science – 2004". – Baku-Vinnytsia-Veliko Turnovo, 2004. – V. 2. – P. 577-579.
13. Borger E., Gradel E., Gurevich Y. The Classical Decision Problem. Perspectives inMathematical Logic.Springer-Verlag, 1997.
14. Baade F., Calvanese D., McGuinness D., Nardi D. and Pater-Schneder P.F, editors. The Description Logic Handbook. Cambridge University Press, 2003.
15. Baader F., Horrocks I., and Sattler U.. Description Logics as Ontology Languages for the Semantic Web. Lecture Notes in Artificial Intelligensce, Volume 2605. Springer-Verlag, 2005.
16. Baader F., Sattler U.. Expressive Number Restrictions in Descriction Logics. Journal of Logic and Compution, 1999. – 9(3). – P. 319-350.
17. Giorgos Stoilos, Giorgos Stamou and Stefanos Kollias A String Metric For Ontology Alignment // In Yolanda Gil, Enrico Motta, Richard Benjamins and Mark Musen, editors, Proceedings of the 4rd International Semantic Web Conference (ISWC), volume 3729 of LNCS, pages 624–637, Berlin. Springer, 2005.
18. Gross O., Wagner R. Continuous Colonel Blotto Game. – RAND Corporation RM-408, 1950. – 13 p.
19. Gruber T. R. Translation approach to portable ontologies / T. R. Gruber // Knowledge Acquisition. – 1993. – N 5 (2). – P. 199 – 220.
20. Gruber T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing / T. R. Gruber // International Journal Human-Computer Studies. – 1995. – № 43(5-6). – P. 907–928.

21. Guilin Qi, Qiu Ji, Peter Haase Combination of Similarity Measures in Ontology Matching using the OWA Operator // In Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Base Systems (IPMU'08), 2008.
22. Guiseppe De Giacomo and Maurizio Lenzerini, Daniele Nardi and Riccardo Rosati. Description Logic Framework for Information Integration. In Proc. of KR'98, pp. 2-13, Morgan Kaufmann, San Francisco, 1998.
23. Guiseppe De Giacomo and Maurizio Lenzerini. What's in an Aggregate: Foundations for Description Logics with Tuples and Sets. In Proc. of IJCAI'95, P.801-807. Morgan Kaufmann, 1995.
24. Guitouni A., Wheaton K. and Wood D. An Essay to Characterise Models of the Military Decision// Making Processes, 11th ICCRTS. – 2006. – Sep 06.
25. Horrocks, U.Sattler, S.Tobies. Practical Reasoning for Very Expressive Description Logics. Logic Journal of the IGPL (Interest Group in Pure Logic), 8(3):239-263, 2000.
26. Horrocks U., Sattler A. Tableaux Desition Prosedure for SHOI2. InProseedings of 19th International Joint Conference on Artificial Intelligence (IJCAI'2005), 2005.
27. Ian Horrocks, Ulrike Sattler, Sergio Tessaris, and Stephan Tobies. Query Containment Using a DLR ABox. LTCS-Report 99-15, LuFG Theoretical Computer cience, RWTH Aachen, Germany, 1999.
28. ISO/IEC 9126:1991. Information technology – Software product evaluation – Quality characteristics and guidelines for their use, 1991. – 39 p.
29. Euzenat J. An API for Ontology Alignment // In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, Proceedings of the 3rd International Semantic Web Conference, vol. 3298 of LNCS, P. 698–712, Berlin. Springer 2004.

30. Bontcheva K. and Sabos M. (2006) Learning Ontologies from Software Artifacts: Exploring and Combining Multiple Sources // In Workshop on Semantic Web Enabled Software Engineering (SWESE), Athens, G.A., USA.
31. Karkulovskyi B., Oborska, O., Teslyuk, V. Hardware-software complex for examining the characteristics of micro-accelerometers (2013) 2013 12th International Conference: The Experience of Designing and Application of CAD Systems in Microelectronics, CADSM, 2013. – P. 445-446.
32. Knappe R., Bulskov H., Andreasen T. (2004) Perspectives on Ontology-based Querying // International Journal of Intelligent Systems. – http://akira.ruc.dk/~knappe/publications/ijis2004.pdf
33. Linckels, Christoph Meinel (2007) Semantic interpretation of natural language user input to improve search in multimedia knowledge base // Information Technologies. – 49(1). –P 40-48.
34. Lytle R. Information Resource Management: 1981–1986. Annual Review ofInformation Science and Technology, 21, 1986. – P. 309-335.
35. Lytvyn V. Design of intelligent decision support systems using ontological approach // An international quarterly journal on economics in technology, new technologies and modelling processes. – Lublin-Lviv. – 2013. – Vol. II. – No 1. P. 31-38.
36. Lytvyn V. Intelligent agent on the basis of adaptive ontologies construction [Електронний ресурс] / V. Lytvyn, D. Dosyn, M. Medykovskyj, N. Shakhovska // XIV International Conference «System Modelling and Control», Lodz, Poland, 27-29 June 2011. – 1 електрон. опт. диск (CD-ROM). – Назва з титул. екрану.
37. Lytvyn V. Method development and qualiti evaluation of an ontology / Vasyl Lytvyn, Dmytro Dosyn, Roman Vovnjanka, Maria Hopyak, Oksana Oborska // XIIIth International Conference. The Experience of Designing and Application of CAD Systems in Microelectronics: – Polyana-Svalyava (Zakarpattya), Ukraine: 24-27 February 2015. – P. 113-115.

38. Lytvyn V. Method of automated development and evaluation of ontologie's qualities of knowledge / V. Lytvyn, M. Hopyak, O. Oborska // Instytut Technologicznych Systemów Informacyjnych. Politechnika Lubelska. Journal "Applied-computer-science". – Poland, 2014. vol. 10 - No 4. – P. 26-37.
39. Lytvyn V., Semotuyk O., Moroz O. Definition of the semantic metrics on the basis of thesaurus of subject area // An international quarterly journal on economics in technology, new technologies and modelling processes. – Lublin-Lviv, 2013. – vol. II. – № 4. – P. 47-51.
40. Lytvyn V., Dosyn D., Smolarz A. An ontology based intelligent diagnostic systems of steel corrosion protection // Elektronika, 2013. – Poland. – № 8. – P. 22-24,
41. Lytvyn V.V. Intelligent Agents Based on Adaptive Ontology / V.V. Lytvyn, O.V. Oborska // IV Міжнародній науковій конференції студентів, аспірантів та молодих вчених. Теоретичні та прикладні аспекти кібернетики (ТААС-2014): Україна. – Київ: 24-28 листопада 2014. – С. 264-273.
42. Lytvyn V. V. The similarity metric of scientific papers summaries on the basis of adaptive ontologies / V. V. Lytvyn // Proceedings of VIIth International Conference on Perspective Technologies and Methods in MEMS Design, Polyana, Ukraine, 11-14 May 2011. – Lviv : IEEE; LPNU, 2011. – 162 p.
43. Lytvyn V. Modelling of intellectual agent behavioral plan based onPetri nets and ontology approach / V. Lytvyn, D. Dosyn, R. Darevych // Computerscience and information technologies : proc. of the V Intern. sci. and techn.conf. CSIT 2010, 14-16 Oct. 2010, Lviv, Ukraine / Lviv Polytechnic Nat. Univ. – Lviv : Publ. House Vezha and Co, 2010. – P. 149-150. – Bibliogr.: 4titles.
44. Miller G. A. WORDNET: A lexical database for English / G.A. Miller // Communications of ACM. – 1995. – N 11. – P. 39 – 41.

45. Necib C. B. Ontology based Query Processing in Database Management Systems / C. Necib, J. Freytag // Proceeding on the 6 th international on ODBASE. – 2003. – P. 37–99.
46. Oborska O. The problem of building automated ontology base / O.Oborska, S.Khrushch // 9th International Scientific and Technical Conference. Computer Sciences and Information Technologies (CSIT'2014): – at Lviv Polytechnic National University: 18-22 November 2014. – C. 113.
47. Blackburn P., Banthem J., Wolter F., editors. Handbook of Modal Logic. Studies in Logic and Practical Reasoning, Volume 3, Elsevier, 2007.
48. Richards C. Certain To Win: The Strategy Of John Boyd, Applied To Business / C.Richards. – Philadelphia: Xlibris Corporation, 2004.
49. Shakhovska N. Web-community ontological representation using intelligent dataspace analyzing agent / N. Shakhovska. , Y. Syerov// Proc. of the X-th Intern. Conf. CADSM-2009 «The Experience of Designing and Application of CAD Systems in Microeletronics». – Polyana–Svaliava (Zakarpattya), 2009. – P. 479–480.
50. Gruber T. A. Translation approach to portable ontologies // Knowledge Acquisition, 1993. – № 5 (2). – P. 199-220.
51. Andreasen T., Bulskov H., and Knappe R. Perspectives on Ontology-based Querying // International Journal of Intelligent Systems. Режим доступу: http://akira.ruc.dk/~knappe/publications/ijis2004.pdf.
52. Gruber T. R. Toward principles for the design of ontologies used for knowledge sharing. Presented at the Padua workshop on Formal Ontology, March 1993, later published in International Journal of Human-Computer Studies, Vol. 43, Issues 4-5, November 1995, P. 907-928 .
53. Teslyuk V. M., Kis Y. P., Teslyuk T.M. Improving the efficiency of the solution of linear programming tasks with usage of CUDA-technology // Actual Problems of Economics, 2014. – № 1(151). – P. 536-541.

54. Uschold M. Ontologies: principles, methods, and applications / M. Uschold, M. Gruninger // Knowledge Engineering Review. – 11(2). – 1996. – P. 93-155.
55. Lytvyn V., Shakhovska N., Pasichnyk V., Dosyn D. Searching the Relevant Precedents in Dataspaces Based on Adaptive Ontology // Computational Problems of Electrical Engineering: International journal.– Lviv, 2012.– Vol.2, No1.– P. 75-81.
56. Vasyliuk I., Teslyuk V., Zelinslyy A., Seniv M. Decision making process based on ontology in MEMS structure (2013) 2013 12th International Conference: The Experience of Designing and Application of CAD Systems in Microelectronics, CADSM 2013. – P. 373.
57. W3C. OWL: Web Ontology Language [Електронний ресурс]. – Режим доступу: http://www.w3.org/TR/owl-absyn/.
58. Kazakov Y., Sattler U., Zolin E. How many legs do I have? Non-simple roles in number restrictions revisited. In proceesings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'2007), 2007.
59. Ивлев А. А. Онтология военных технологий на основе концептуальных карт / А. А. Ивлев, В. Б. Артеменко // Электронный научный журнал «Исследовано в России». - Режим доступу: http://zhurnal.ape.relarn.ru/articles/2011/023.pdf
60. Андрейчикова О. Н. Интеллектуальные системы для поддержки процессов принятия решений / О.Н.Андрейчикова. – Волгоград: Изд-во ВолгГТУ, 1996. – 93 с.
61. Артемьева И. Л. Метод построения многоуровневых онтологий сложноструктурированных предметных областей [Електронний ресурс] / И.Л. Артемьева // Знания – онтологии – теории: труды Всерос. конф. – Новосибирск, 2007. – Режим доступа: http://www.iacp.dvo.ru/is/ publications/Artemjeva.pdf.

62. Болотова В. А. Инструментальные средства создания баз знаний на основе системы онтологий: автореф. дис. на здобуття наук. ступеня спеціаліст: спец. «Програмне забезпечення автоматизованих систем».
63. Буренок В.М., Ивлев А.А., Корчак В.Ю. Развитие военных технологий XXI века: проблемы, планирование, реализация. – Тверь: Изд. «Купол», 2009. – 624 с.
64. Бусленко Н.П. Моделирование сложных систем. – М.: Наука, 1968. – 356 с.
65. Вентцель Е.С. Введение в исследование операций / Е.С. Вентцель. – М.: «Советское радио», 1964. – 390 с.
66. Гаврилова Т.А. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб. : Питер, 2001. – 384 с.
67. Гладун А.Я. Формирование тезауруса предметной области как средства моделирования информационных потребностей пользователя при поиске в Интернете / А.Я. Гладун, Ю.В. Рогушина // Вестник компьютерных и информационных технологий. – 2007. – № 1. – С. 26-33.
68. Даревич Р.Р. Агентна система автоматичного опрацювання науково-технічної інформації на основі її інтелектуального аналізу / Р.Р. Даревич, Д.Г. Досин // Автоматика-2004: Матер. 11-ї міжнар. конф. з автоматичного управління, м. Київ, 27–30 вересня 2004 року. – К. : НУХТ, 2004. – Т.4. – 35 с.
69. Даревич Р.Р. Використання адаптивних онтологій в інтелектуальних системах прийняття рішень / Р.Р. Даревич, Д.Г. Досин, В.В. Литвин // Відбір і обробка інформації. – 2009. – Вип. 31(107). – С. 118-126.
70. Денисов А.А. "Призрачные" субъекты в управлении современным военным и политическим конфликтом. // 2010.
71. Денисов А.А. Нетократия и рефлексия: Засекречивание в постиндустриальном обществе. // "Рефлексивные процессы и управление". – Том 7. – No. 1. – 2007. – С. 33-50.

72. Денисов А. А. Новая комплексная технология информационно - аналитического обеспечения стратегического управления. // Тезисы доклада. 2-я Всероссийская научно-практическая конференция "Информационно-аналитическое обеспечение стратегического управления: теори.
73. Денисов А. А. Системы, превосходящие исследователя по совершенству . // IV Международная конференция по проблемам управления. Сборник трудов. // М., Учреждение РАН Институт проблем управления им. В. А. Трапезникова РАН, 2009. – С. 1356-1363.
74. Денисов А. А., Денисова Е. В. Постиндустриализм: проблемы и задачи новой кадровой политики. // "Экономические стратегии ", No.3 (69), 2009. – С. 64-71.
75. Джорждд Л. М. «Бережливое производство+ шесть сигм» в сфере услуг: Как скорость бережливого производства и качество шести сигм помогают совершенствованию бизнеса//. – М.: Альпина Бизнес Букс– 2005. – 402 с.
76. Досин Д. Г. Моделювання поведінки інтелектуального агента на основі онтологічного підходу / Д. Г. Досин, В. В. Литвин, Н. В. Шкутяк // Відбір і обробка інформації. – 2009. – Вип. 31(107). – С. 112. - 117.
77. Емельянов А. А. Имитационное моделирование экономических процессов: учебн. пособ. / А. А. Емельянов, Е. А. Власова, Р. В. Дум. – М. : Финансы и статистика, 2002. – 368 с.
78. Ефимов Е. И. Решатели интеллектуальных задач / Е.И. Ефимов. – М.: Наука, 1982. – 316 с.
79. Живчук В. Л. Розробка робочого місця розвідника в складі автоматизованої системи управління СВ ЗСУ / В. Л. Живчук, В. В. Литвин, О. В. Оборська // Міжнародна науково-технічна конференція "Перспективи розвитку озброєння та військової техніки Сухопутних військ". – Україна, Львів: 14-15 травня 2015. – С. 142.

80. Згуровський М. З., Петренко А. І. Е-наука на шляху до семантичного Грід. Частина 2: семантичний web- і семантичний грід // Системні дослідження та інформаційні технології. – 2010. – № 2. – С. 26-38.
81. Ивлев А. А. Основы теории Бойда. Направления развития, применения и реализации.// М., 2008.
82. Искусственный интеллект. – В 3-х кн. Кн. 2. Модели и методы: справочник / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – 304 с.
83. Інтелектуальні системи, базовані на онтологіях // Д. Г. Досин, В. В. Литвин, Ю. В. Нікольський, В. В. Пасічник. – Львів: “Цивілізація”, 2009. – 414 с.
84. Інтелектуальні системи: підручник / Василь Володимирович Литвин, Володимир Володимирович Пасічник, Юрій Володимирович Яцишин . — Львів :Новий Світ-2000, 2011. — С. 405: іл. — (Комп’ютинг) . — Бібліогр.: С. 384-402(322 назви) . —ISBN 978-966-418-086-0.
85. Казмірчук Р. В. Світовий досвід і тенденції застосування засобів імітаційного моделювання бойових дій / Р.В. Казмірчук, Є.В. Рижов, О.В. Корольова, В. І. Боженко // «Військово-технічний збірник». – 2009. – Режим доступу: http://www.asv.gov.ua/content/na.
86. Кара-Мурза С. Подрыв рационального мышления и рефлексивное управление. // "Рефлексивные процессы и управление". - Том 3. – No. 2. – 2003, С. 16-34.
87. Катренко А. В. Дослідження операцій.– Львів: Магнолія Плюс, 2004. – 549с.
88. Кириченко С. О. Система управління Збройних Сил України: ретроспективний аналіз і перспективи розвитку // Наука і оборона. – 2007. – № 3. – С. 13-18.
89. Клейнер Д. Статистические методы в имитационном моделировании / Дж. Клейнер. – М. : Статистика, 1978. – 256 с.

90. Клещёв А. С. Отношения между онтологиями предметных областей. Ч.1. / А.С. Клещёв, И.Л. Артемьева // Информационный анализ. – Выпуск 1. – Се Искусственный интеллект. – В 3-х кн. Кн. 1. Системы общения и экспертные системы: справочник / Под ред. Д. А. Поспелова. – М.: Радио и связь, 1990. – 464 с.
91. Ковалевич В. М. Проектування системи автоматизованого синтезу онтології матеріалознавства /В. М. Ковалевич, А. О. Яценко, О. В. Оборська // Міжнародна науково - практична конференція молодих вчених та студентів. Інформаційні технології, економіка та право: стан та перспективи розвитку (ІТЕП-2014): – Чернівці: 3-4 квітня 2014. – С. 30-31.
92. Коваленко С. П. Метод ефективного розподілу цілей при управлінні вогнем підрозділу. Системи обробки інформації / С. П. Коваленко, О. В. Коломійцев, В. В. Обрядін, К.І. Худа- рковський // Системи обробка інформації. – Х.: ХУПС. – 2007. – Вип. 3(61). – С. 41-43.
93. Комарова Л. О. Дослідження функцій ураження об'єктів різних класів для задач цілерозподілу / Л. О. Комарова, А. І. Невольніченко / Мат. машини і системи. – 2012. — № 4. – С. 156-167.
94. Крайовський В. Я. Використання адаптивних онтологій в інтелектуальних системах прийняття рішень / В. Я. Крайовський, В. В. Литвин, Н. Б. Шаховська // Східноєвропейський журнал передових технологій. – №4/3(40). – Харків, 2009. – С.7-12.
95. Лефевр В. Системы, сравнимые с исследователем по совершенству. // Рефлексия. – М., "Когито-центр", 2003.
96. Литвин В. В. Автоматизація процесу розвитку базової онтології на основі аналізу текстових ресурсів / В. В. Литвин // Інформаційні системи та мережі : [зб. наук. пр.] / відп. ред. В. В. Пасічник. — Л. : Вид-во Львів. політехніки, 2010. – С. 319-325.

97. Литвин В. В. Аналіз методів розв’язування задачі планування в обчислювальних грід-системах / В. В. Литвин, А. С. Мельник // Інформаційні системи та мережі : [зб. наук. пр.] / відп. ред. В.В. Пасічник. — Л. : Вид-во Львів. політехніки, 2010. – С. 189-200.
98. Литвин В. В. Бази знань інтелектуальних систем підтримки прийняття рішень / Литвин В. В. – Львів: Видавництво Львівської політехніки, 2011. – 240 с.
99. Литвин В.В. Інтелектуальні агенти пошуку релевантних прецедентів на основі адаптивних онтологій / В. В. Литвин // Математичні машини і системи. – 2011. – № 3. – С. 66-72.
100. Литвин В.В. Метод використання онтологій в петлі OODA для моделювання воєнних дій / В. В. Литвин, Л. Л. Джавала, Е. В. Лучук // Системи обробки інформації, 2013, випуск 4 (111). – С. 76-81.
101. Литвин В. В. Метод моделювання процесу підтримки прийняття рішень у конкурентному середовищі / В. В. Литвин, О. В. Оборська, Р. В. Вовнянка // Математичні машини й системи. Науковий журнал Інституту проблем математичних машин і систем НАН України. – Київ, 2014. №1. – С.50-57.
102. Литвин В. В. Метод підтримки прийняття рішень у конкурентному середовищі на основі петлі OODA. /В. В. Литвин, М. Я. Гопяк, Р. В. Вовняна, О. В. Оборська // Матеріали міжнародної наукової конференції: Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: – Залізний Порт. – Херсон: ХНТУ, 2014. – С. 219-220.
103. Литвин В.В. Метод планування рішень у конкурентному середовищі на основі використання онтологій / В. В. Литвин, О. В. Оборська, М. Я. Гопяк, Р. В. Вовнянка // Десята міжнародна науково-практична конференція. Математичне та імітаційне моделювання систем (МОДС 2015): – Україна, Чернігів: 22-26 червня 2015. – С. 460-465.

104. Литвин В. В. Моделювання автоматизованої систем управління тактичної ланки на основі онтологічного підходу / В. В. Литвин, О. В. Оборська // Вісник Кременчуцького національного університету імені Михайла Остроградського. – Кременчук, 2014. – Випуск 5/2014(88). – С. 92-97.
105. Литвин В .В. Моделювання діяльності раціонального агента на основі адаптивних онтологій / В. В. Литвин, В. Я. Крайовський // 4-та Міжнар. наук.-техн. конф. “Комп’ютерні науки та інформаційні технології 2009”. – С. 308–310.
106. Литвин В. В. Моделювання етапу «Орієнтація» в петлі OODA під час інформаційної війни / В. В. Литвин, О. В. Оборська // Матеріали 2-ї міжнародної наукової конференції (ІКС-2014). Інформація, комунікація, суспільство: – Україна, Львів-Славське: 21-24 травня 2014. – С. 190-191.
107. Литвин В. В. Моделювання інтелектуальних систем підтримки прийняття рішень з використанням онтологічного підходу / В. В. Литвин // Радіоелектроніка, інформатика, управління / Запорізький національний технічний університет. – 2011. – № 2 (25). – С. 93-101.
108. Литвин В .В. Моделювання плану поведінки інтелектуального агента на основі мереж Петрі та онтологічного підходу / В. В. Литвин // Інформаційні системи та мережі: Вісник Нац. ун-ту “Львівська політехніка”. – 2009. – № 653. – С. 170-175.
109. Литвин В. В. Моделювання поведінки інтелектуального агента на основі петлі OODA / В. В. Литвин, М. Я. Гопяк Р. В. Вовняна О. В. Оборська // Міжнародна науково-практична конференція. Інформаційні технології.Освіта: – Україна, Луцьк-Світязь: 6-8 червня 2014. – С. 60-61.
110. Литвин В. В. Моделювання процесу підтримки прийняття рішень в конкурентному середовищі на основі петлі OODA / В. В. Литвин, Р. В. Вовнянка, О. В. Оборська // Міжнародна науково-практична конференція. Інформаційні технології.Освіта: – Україна, Луцьк-Світязь: 3-4 червня 2013. – С. 31-32.

111. Литвин В. В. Онтологічний підхід до побудови АСУ тактичної ланки / В. В. Литвин, О. В. Оборська // ІІІ науково-технічна конференція Фізико-механічний інститут ім. Г.В. Карпенка НАН України. Обчислювальні методи і системи перетворення інформації: – Україна, Львів: 25-26 вересня 2014. – С. 188-191.
112. Литвин В. В. Оцінка новизни знань під час автоматичної розбудови онтологій / В. В. Литвин, А. С. Мельник, В. Я. Крайовський // Інформаційні системи та мережі. Вісник НУ "Львівська політехніка". – 2011. – № 699. – С. 343-353.
113. Литвин В. В. Підхід до побудови програмного забезпечення збору, передачі та обробки розвідувальних даних / В. В. Литвин, В. Л. Живчук, О. В. Оборська // Військово-науковий вісник. – 2015. – Випуск 2(9). – С. 43-46.
114. Литвин В. В. Проектування інтелектуальних агентів на основі адаптивних онтологій / В. В. Литвин, Н. Б. Шаховська, А. С. Мельник, О. Ю. Пшеничний, Ю. В. Ришковець // Міжнар. наук. конф. "Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту" ISDMCI'2010. – Євпаторія. – Т.2. – С. 401-404.
115. Литвин В. В. Проектування інтелектуальних агентів прийняття рішень в просторі ознак з використанням онтологічного підходу / В. В. Литвин, Р. Р. Даревич, Д. Г. Досин, Н. В. Шкутяк // Штучний інтелект. – Донецьк–Кацивелі. – 2010. –Т. 2. – С. 100-104.
116. Литвин В. В. Проектування інформаційних систем: навчальний посібник / В. В. Литвин, Н. Б. Шаховська – Львів: «Магнолія-2006», 2011. – 380 с.
117. Литвин В. В. Система підтримки прийняття рішень як складова автоматизованої системи управління сухопутних військ збройних сил України / В. В. Литвин, Е. В. Лучук, П. П. Ткачук // Військово-науковий вісник. – 2013. – Випуск 2(9). – С. 43-46.

118. Лучук Е. В. Загальні принципи побудови автоматизованої системи управління тактичої ланки сухопутних військ ЗСУ. Геоінформаційні системи та інформаційні технології у військових і спеціальних задачах: Збірка матеріалів науково-технічного семінару. – Львів: АСВ, 2014. – С. 14-20.
119. Люгер Дж. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем / Дж. Ф. Люгер, 4-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2005. – 864 с.
120. Мазурін О. Комп’ютерні баталії / О. Мазурін // Військо України. – 2006. – № 1. – С.35-38.
121. Майер К., Дэвис С. Живая организация// М.: Издательство «Добрая книга» – 2007 – 368 с.
122. Макаси Имаи Кайдзен: ключ к успеху японских компаний// М.: Альпика Бизнес Букс– 2006. – 274 с.
123. Максимей И. В. Имитационное моделирование на ЭВМ / И. В. Максимей. – М. : Радио и связь, 1988. – 232 с.
124. Математические модели боевых действий / П. Н. Ткаченко, Л. Н. Куцев, Г. А. Мещеряков, А. М. Чавкин, А. Д. Чебыкин. – М.: Советское радио, 1969. – 171 с.
125. Маурер Р. Путь Кайдзен //. Минск.: ООО «Попури» – 2005. – 192 с.
126. Мацяшек Л. Анализ и проектирование информационных систем с помощью UML 2.0 / Л. Мацяшек. – М.: ООО "И.Д. Вильямс", 2008. – 816 с.
127. Методи та засоби інженерії даних та знань [Текст] : навч.посібник / В. В. Литвин. – Львів :Магнолія-2006, 2012. – 240 с. – (Комп'ютинг). – ISBN 978-617-574-044-6
128. Можаровський В. М. Основні положення методики визначення варіанта (способу) бойових дій та складу угрупування військ (сил) для відбиття

агресії / В. М. Можаровський, О. М.Загорка // Наука і оборона / №1’2011. – С. 3-6.

129. Муромцев Д. И. Онтологический инжиниринг знаний в системе Protégé: Методическое пособие. — СПб: СПбГУ ИТМО, 2007. – 62 с. http://window.edu.ru/window_catalog/redir?id=54429&file=itmo240.pdf
130. Найханова Л. В. Технология создания методов автоматического построения онтологий с применением генетического и автоматного программирования – Улан-Удэ: Изд-во БНЦ СО РАН, 2008. – 244 с.
131. Нариньяни, А. С. ТЕОН-2: от тезауруса к онтологии и обратно / А. С. Нариньяни // Компьютерная лингвистика и интеллектуальные технологии: междунар. семинар Диалог’2002. – М.: Наука, 2002. – Т. 1. – С. 307-313.
132. Новиков Ф. А. Дискретная математика для программистов / Новиков Ф. А. – СПб.: Питер, 2004. – [2-е изд.]. – 364 с.
133. Норенков И. П. Информационно-образовательные среды на базе онтологического похода / И. П. Норенков, М. Ю. Уваров // В сборнике научных статей "Интернет-порталы: содержание и технологии". Выпуск 3. / Редкол.: А. Н. Тихонов (пред.) и др.; ФГУ ГНИИ ИТТ "Информика". – М. : Просвещение, 2005. – С. 367-378.
134. Оборська О. В. Інформаційна система моделювання воєнних дій механізованих військ з використанням онтологічного підходу / О. В. Оборська // Проблеми програмування. – 2014. – № 4. – С. 59-66.
135. Оборська О. В. Моделювання етапу «Орієнтація» в петлі OODA під час інформаційної війни / В. В Литвин, О. В. Оборська // матеріали II Міжнародної наукової конференції "Інформація, комунікація, суспільство" (ІКС – 2014). (21–24 травня 2014, Львів). С. 190-19
136. Оборська О. В. Моделювання поведінки раціонального агента на основі стимулюючого навчання / О. В.Оборська, Р. В.Вовнянка // Інформаційні

системи та мережі. Вісник Національного університету “Львівська політехніка”. – Львів, 2014. – №805. – С. 61-69.

137. Оборська О. В. Онтологічний підхід до побудови АСУ тактичної ланки / В. В. Литвин, О. В. Оборська // III науково-технічна конференція "Обчислювальні методи і системи перетворення інформації" Фізико-механічний інститут ім. Г. В. Карпенка, 2014. – С. 188-192.
138. Оборська О. В. Розробка модуля імітаційного моделювання бойових дій для етапу „орієнтація” циклу OODA / О. В. Оборська, Р. В. Вовнянка, М. Я. Гопяк // IV-й Міжнародній науковій конференції ІКС-2015 “Інформація, комунікація, суспільство”: – Україна, Львів-Славське: 20-23 травня 2015. – С. 50-52.
139. Палагин А. В. Архитектура онтологоуправляемых компютерных систем / А. В. Палагин // Кибернетика и системный анализ. – № 2. – 2006. – С. 111-125.
140. Пасічник В. Інформаційні технології та системи дистанційного навчання осіб з особливими потребами / В. Пасічник, В. Кут // Вісник ТНТУ. – 2012. – Том 65. – № 1. – С. 127-137. – (приладобудування та іформаційно-вимірювальні технології).
141. Пермяков О. Ю. Шляхи інтегрування імітаційного моделювання у процес оперативної і бойової підготовки Збройних Сил України / О.Ю. Пермяков // Тези доповіді на кафедрі інформатизації штабів. – К.: НАОУ, 2006. – С. 17-22.
142. Поспелов Д. А. Моделирование рассуждений. Опыт анализа мыслительных актов / Д. А. Поспелов. – М. : Радио и связь, 1989. – 184 с.
143. Постаногов Д. Ю. К вопросу многоязычности систем инженерии знаний и их приложений / Д. Ю. Постаногов, И. В. Совпель // Искусственный интеллект. – 2006. – Вип. 3. – С. 474-479.
144. Рассел С. Искусственный интеллект / С. Рассел, П. Норвиг. – М.; СПб.; К.: Вильямс, 2006. – 1408 с.

145. Резяпов Н. Развитие систем компьютерного моделирования в вооруженных силах США / Н. Резяпов// Зарубежное военное обозрение. – 2007. – №6. – С. 17-23.
146. Рогушина Ю. В. Використання онтології як засобу інтеґрації знань про інформаційну систему / Ю.В. Рогушина, А.Я. Гладун // Відбір і обробка інформації. – №24(100). – 2006. – С. 43-48.
147. Рувинская В. М. Мультиагентные системы для поиска информации в Интернете с использованием онтологий / В. М. Рувинская, К. Л. Манукян // “Искуственный интелект”, ІПШІ “Наука і освіта”, 2002. – № 4. – С. 606-613.
148. Сетлак Г. Решение задач многокритериальной оптимизации с использованием генетических алгоритмов [Текст] // System Research and Information Technologies. Kiev: IASA National Academy of Sciences and Ministry of Education and Science Ukraine. 2002. – № 3. – P. 32-42.
149. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: Навчальний посібник. — Запоріжжя: ЗНТУ, 2008. – 341 с.
150. Ткаченко П. Н. Математические модели боевых действий / П.Н. Ткаченко, Л. Н. Куцев, Г. А. Мещеряков, А. М. Чавкин, А. Б. Чебыкин // Москва: Издательство «Советское радио», 1969р. – С. 350-365.
151. Шеннон Р. Имитационное моделирование систем – искусство и наука / Р. Шеннон. – М.: Мир, 1978. – 420 с.
152. Перегуда О. М., Піонтківський П. М., Поліновський В. М., Шестаков В. І. Концепція створення та архітектура єдиного інформаційно-обчислювального середовища системи кризового управління / О. М. Перегуда, П. М. Піонтківський, В. М. Поліновський, В. І. Шестаков // Наукова стаття. Проблеми створення, випробування, застосування та

експлуатації складних інформаційних систем: Збірник наукових праць ЖВІ. – Житомир: ЖВІ, 2015. – Вип. 12. – С. 51-59.

ДОДАТОК А

Фрагмент коду застосунку «Military intelligence»

MainActivity.java

```
package in.wptrafficanalyzer.locationmarkerpreferences;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OptionalDataException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.GoogleMap.OnMapClickListener;
import com.google.android.gms.maps.GoogleMap.OnMarkerClickListener;
import com.google.android.gms.maps.GoogleMap.OnMarkerDragListener;
```

```
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

public class MainActivity extends FragmentActivity implements
    OnMarkerClickListener, Serializable {

  GoogleMap googleMap;
  Marker marker;
  JSONParser jsonParser = new JSONParser();
  private ProgressDialog pDialog;

  int count = 0;

  SharedPreferences sharedPreferences;
  int locationCount = 0;
  private static String url_add_record = "http://192.168.1.101/android/add_record.php";
  private HashMap<String, ObjectInfo> hashMap = new HashMap<String, ObjectInfo>();
  HashMap<String, ObjectInfo> eventMarkerMap = new HashMap<String, ObjectInfo>();
  // FileOutputStream fileOutputStream;

  EditText incomment, inusername, inx, iny, inobject, inmodel, date, time;
  LatLng temp = null;
  double lat, lon;
  int adress, backButtonCount = 0;
  String id;
  ArrayList<Marker> list = new ArrayList<Marker>();

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    this.closeOptionsMenu();

    int status = GooglePlayServicesUtil
        .isGooglePlayServicesAvailable(getBaseContext());

    if (status != ConnectionResult.SUCCESS) {

      int requestCode = 10;
      Dialog dialog = GooglePlayServicesUtil.getErrorDialog(status, this,
          requestCode);
      dialog.show();

    } else {

      SupportMapFragment fm = (SupportMapFragment) getSupportFragmentManager()
          .findFragmentById(R.id.map);
```

```
    googleMap = fm.getMap();

    googleMap.setMyLocationEnabled(true);

  }

  googleMap.setOnMarkerClickListener(this);

  // встановлення лістенера для об'ктів карти
  googleMap.setOnMarkerDragListener(new OnMarkerDragListener() {

    public void onMarkerDragStart(Marker marker) {
      marker.remove();
      hashMap.remove(marker.getId());
      Toast.makeText(getBaseContext(), "Об'єкт видалено",
          Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onMarkerDragEnd(Marker marker) {
      marker.remove();
      hashMap.remove(marker.getId());
      Toast.makeText(getBaseContext(), "Об'єкт видалено",
          Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onMarkerDrag(Marker marker) {
      marker.remove();
      hashMap.remove(marker.getId());
      Toast.makeText(getBaseContext(), "Об'єкт видалено",
          Toast.LENGTH_SHORT).show();
    }
  });

  googleMap.setOnMapClickListener(new OnMapClickListener() {

    @Override
    public void onMapClick(LatLng point) {
      locationCount++;
      drawMarker(point);

    }
  });
}

// метод для реалізації дії кнопки виходу з програми
@Override
public void onBackPressed() {
  if (backButtonCount >= 1) {
```

```
      super.onBackPressed();
      Intent intent = new Intent(Intent.ACTION_MAIN);
      intent.addCategory(Intent.CATEGORY_HOME);
      intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
      startActivity(intent);

    } else {
      Toast.makeText(this, "Натисніть ще раз, щоб вийти",
          Toast.LENGTH_SHORT).show();
      backButtonCount++;
    }
}

@Override
public void onPause() {
  super.onPause();
}

@Override
public void onStop() {
  super.onStop();
}

// метод встановлення об'єкта на карту у вибрану позицію
private void drawMarker(LatLng point) {

  MarkerOptions markerOptions = new MarkerOptions();
  markerOptions.position(point).draggable(true);
  marker = googleMap.addMarker(markerOptions);
  Toast.makeText(getBaseContext(), "Об'єкт додано на карту",
      Toast.LENGTH_SHORT).show();

}

// реалізація контекстного меню програми
@Override
public boolean onCreateOptionsMenu(Menu menu) {

  getMenuInflater().inflate(R.menu.main, menu);
  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
  case R.id.action_settings:
    // if(isOnline()) {
    new CreateNewRecord().execute();
    // }
    // else {
    // try {
```

```
      // saveData();
      // Toast.makeText(getBaseContext(), "Дані збережено в пам'яті",
      // Toast.LENGTH_SHORT).show();
      // } catch (FileNotFoundException e) {
      // e.printStackTrace();
      // } catch (IOException e) {
      // e.printStackTrace();
      // }
      // }
      break;
    case R.id.clear:
      alertMessage();
      break;
    case R.id.ip:
      inputIp();
      break;

    case R.id.video:
      // Intent i;
      // PackageManager manager = getPackageManager();
      // try {
      // i =
      // manager.getLaunchIntentForPackage("com.example.javacv.stream.test2");
      // if (i == null)
      // throw new PackageManager.NameNotFoundException();
      // i.addCategory(Intent.CATEGORY_LAUNCHER);
      // startActivity(i);
      // } catch (PackageManager.NameNotFoundException e) {
      // }
      break;
    }
    return super.onOptionsItemSelected(item);
  }

  public void saveData() throws FileNotFoundException, IOException {
    FileOutputStream fileOutputStream = openFileOutput("data",
        Context.MODE_PRIVATE);
    ObjectOutputStream objectOutputStream = new ObjectOutputStream(
        fileOutputStream);
    objectOutputStream.writeObject(hashMap);
    objectOutputStream.close();
  }

  public void loadData() {
    try {
      FileInputStream fileInputStream = openFileInput("data");
      ObjectInputStream objectInputStream = new ObjectInputStream(
          fileInputStream);
      HashMap map = (HashMap) objectInputStream.readObject();
    } catch (OptionalDataException e) {
      // TODO Auto-generated catch block
```

```
        e.printStackTrace();
      } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
    }

    // діалог пітвердження очищення еарти від об'єктів
    public void alertMessage() {
      DialogInterface.OnClickListener dialogClickListener = new DialogInterface.OnClickListener()
{
        public void onClick(DialogInterface dialog, int which) {
          switch (which) {
          case DialogInterface.BUTTON_POSITIVE:
            googleMap.clear();
            eventMarkerMap.clear();
            break;
          case DialogInterface.BUTTON_NEGATIVE:
            dialog.cancel();
            break;
          }
        }
      };
      AlertDialog.Builder builder = new AlertDialog.Builder(this);
      builder.setMessage("Ви справді хочете очистити карту?")
          .setPositiveButton("Так", dialogClickListener)
          .setNegativeButton("Ні", dialogClickListener).show();
    }

    public void inputIp() {
      AlertDialog.Builder alert = new AlertDialog.Builder(this);

      alert.setTitle("IP");
      alert.setMessage("Input your ip:");

      // Set an EditText view to get user input
      final EditText input = new EditText(this);
      alert.setView(input);

      alert.setPositiveButton("Ок", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
          String value = input.getText().toString();
          url_add_record = "http://" + value + "/android/add_record.php";
          // Do something with value!
        }
      });

      alert.setNegativeButton("Вихід", new DialogInterface.OnClickListener() {
```

```
        public void onClick(DialogInterface dialog, int whichButton) {
          // Canceled.
        }
      });

      alert.show();
    }

    public boolean isOnline() {
      ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
      NetworkInfo netInfo = cm.getActiveNetworkInfo();
      if (netInfo != null && netInfo.isConnectedOrConnecting()) {
        return true;
      }
      return false;
    }

    // перевизначення лістенера для об'єкта
    @Override
    public boolean onMarkerClick(final Marker marker) {

      temp = marker.getPosition();
      lat = temp.latitude;
      lon = temp.longitude;
      id = marker.getId();
      Intent intent = new Intent(this, DialogActivity.class);
      intent.putExtra("lat", lat);
      intent.putExtra("lon", lon);
      intent.putExtra("id", id);
      intent.putExtra("hashMap", hashMap);

      if (hashMap.containsKey(id)) {
        intent.putExtra("permission", true);
      }
      startActivityForResult(intent, 1);

      return false;

    }

    @SuppressWarnings("unchecked")
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
      String icon = null,s=null;
      eventMarkerMap = (HashMap<String, ObjectInfo>) data
          .getSerializableExtra("eventMarkerMap");
      hashMap.putAll(eventMarkerMap);

      try{
      icon = data.getStringExtra("icon");
      s = icon;
```

```
//    // whose = data.getStringExtra("whose");
      System.out.println(icon);
      changeIcon(s);
      }
      catch(Throwable e) {
        //changeIcon(s);
      }
    }

    // метод зміни іконки об'єкта
    protected void changeIcon(String icon) {
      marker.setIcon(BitmapDescriptorFactory.fromAsset("t" + icon + ".png"));
      // fromResource(R.drawable.t021_03_1_111));

    }

    // Клас створення нового запису в базу даних
    class CreateNewRecord extends AsyncTask<String, String, String> {

      @Override
      protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(MainActivity.this);
        pDialog.setMessage("Додаєм дані...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);
        pDialog.show();
      }

      @Override
      protected String doInBackground(String... args) {

        for (String key : hashMap.keySet()) {

          String typeObj = hashMap.get(key).getTypeObj();
          String object = hashMap.get(key).getObj();
          String move = hashMap.get(key).getMove();
          String angle = String.valueOf(hashMap.get(key).getAngle());
          String levelSafe = hashMap.get(key).getLevelSafe();
          String state = hashMap.get(key).getState();
          String coordinateX = String
              .valueOf(hashMap.get(key).getCoorX());
          String coordinateY = String
              .valueOf(hashMap.get(key).getCoorY());
          String user = hashMap.get(key).getUser();
          String dateS = hashMap.get(key).getDate();
          String timeS = hashMap.get(key).getTime();
          String comments = hashMap.get(key).getComment();
          String whose = hashMap.get(key).getWhose();

          System.out.println(typeObj);
```

```
    System.out.println(object);
    System.out.println(move);
    System.out.println(angle);
    System.out.println(levelSafe);
    System.out.println(state);
    System.out.println(coordinateX);
    System.out.println(coordinateY);
    System.out.println(user);
    System.out.println(dateS);
    System.out.println(timeS);
    System.out.println(comments);
    System.out.println(whose);

    List<NameValuePair> params = new ArrayList<NameValuePair>();
    try {
      params.add(new BasicNameValuePair("typeObj", URLEncoder
          .encode(typeObj, "CP1251")));
      params.add(new BasicNameValuePair("object", URLEncoder
          .encode(object, "CP1251")));
      params.add(new BasicNameValuePair("move", URLEncoder
          .encode(move, "CP1251")));
      params.add(new BasicNameValuePair("angle", URLEncoder
          .encode(angle, "CP1251")));
      params.add(new BasicNameValuePair("levelSafe", URLEncoder
          .encode(levelSafe, "CP1251")));
      params.add(new BasicNameValuePair("state", URLEncoder
          .encode(state, "CP1251")));
      params.add(new BasicNameValuePair("coordinateX", URLEncoder
          .encode(coordinateX, "CP1251")));
      params.add(new BasicNameValuePair("coordinateY", URLEncoder
          .encode(coordinateY, "CP1251")));
      params.add(new BasicNameValuePair("user", URLEncoder
          .encode(user, "CP1251")));
      params.add(new BasicNameValuePair("dateS", URLEncoder
          .encode(dateS, "CP1251")));
      params.add(new BasicNameValuePair("timeS", URLEncoder
          .encode(timeS, "CP1251")));
      params.add(new BasicNameValuePair("comments", URLEncoder
          .encode(comments, "CP1251")));
      params.add(new BasicNameValuePair("whose", URLEncoder
          .encode(whose, "CP1251")));

    } catch (UnsupportedEncodingException e) {
      e.printStackTrace();
    }

    JSONObject json = jsonParser.makeHttpRequest(url_add_record,
        "POST", params);
    Log.d("Create Response", json.toString());

}
```

```
        return null;
      }

      protected void onPostExecute(String file_url) {
        pDialog.dismiss();
      }
    }
}
```

Фрагмент коду реалізації задачі цілерозподілу

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace GenetuchnuyAlgorutm
{
   public partial class Form1 : Form
   {

      public Form1()
      {
         InitializeComponent();
      }
      DataTable dt = new DataTable();
      DataTable dt1 = new DataTable();
      DataTable dt2 = new DataTable();
      DataTable dt3 = new DataTable();
      int i, j;


      private void tableBindingNavigatorSaveItem_Click(object sender, EventArgs e)
      {
         this.Validate();
         this.tableBindingSource.EndEdit();
         this.tableAdapterManager.UpdateAll(this.genAlgDataSet);

      }

      private void Form1_Load(object sender, EventArgs e)
      {
         // Таблиця_1
         dt.Columns.Add("id", typeof(int));
         dt.Columns.Add("Zasib", typeof(string));
```

```
            dt.Columns.Add("K-st", typeof(int));
            dt.Columns.Add("Comentar", typeof(string));

            // Таблиця_1
            dt1.Columns.Add("id", typeof(int));
            dt1.Columns.Add("Cil", typeof(string));
            dt1.Columns.Add("K-st_c", typeof(int));
            dt1.Columns.Add("Vubratu", typeof(bool));

            // Таблиця_2
            dt2.Columns.Add("id", typeof(int));
            dt2.Columns.Add("id_c", typeof(int));
            dt2.Columns.Add("id_z", typeof(int));
            dt2.Columns.Add("Imovirnist", typeof(float));

            // Таблиця_3
            dt3.Columns.Add("id", typeof(int));
            dt3.Columns.Add("id_Cil", typeof(int));
            dt3.Columns.Add("id_Zasib", typeof(int));
            dt3.Columns.Add("Imovirnist_1", typeof(float));

            // TODO: данная строка кода позволяет загрузить данные в таблицу
"genAlgDataSet.Table3". При необходимости она может быть перемещена или удалена.
            this.table3TableAdapter.Fill(this.genAlgDataSet.Table3);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"genAlgDataSet.Table2". При необходимости она может быть перемещена или удалена.
            this.table2TableAdapter.Fill(this.genAlgDataSet.Table2);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"genAlgDataSet.Table1". При необходимости она может быть перемещена или удалена.
            this.table1TableAdapter.Fill(this.genAlgDataSet.Table1);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"genAlgDataSet.Table". При необходимости она может быть перемещена или удалена.
            this.tableTableAdapter.Fill(this.genAlgDataSet.Table);

        }

        private void table2DataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
                        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.table1BindingSource.AddNew();
        }

        private void tableDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
                }
```

```
        private void table1DataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
        }

        private void table3DataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            int ind = tableDataGridView.SelectedCells[0].RowIndex;
            tableDataGridView.Rows.RemoveAt(ind);
        }

      /*   private void button2_Click(object sender, EventArgs e)
        {
            string id = idTextBox.Text;
            string zacib = comboBox1.Text;
            string k_st = k_stTextBox.Text;
            string comentar = comentarTextBox.Text;
            tableDataGridView.Rows.Add(id, zacib, k_st, comentar);
        }
        */
        private void button4_Click(object sender, EventArgs e)
        {
            int ind = table1DataGridView.SelectedCells[0].RowIndex;
            table1DataGridView.Rows.RemoveAt(ind);
        }

        private void button7_Click(object sender, EventArgs e)
        {
            int ind = table2DataGridView.SelectedCells[0].RowIndex;
            table2DataGridView.Rows.RemoveAt(ind);
        }

        private void button9_Click(object sender, EventArgs e)
        {
            int ind = table3DataGridView.SelectedCells[0].RowIndex;
            table3DataGridView.Rows.RemoveAt(ind);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            try
            {

                dt1.Rows.Add(idTextBox1.Text, comboBox2.Text, k_st_cTextBox.Text,
vubratuCheckBox.Checked);
                table1DataGridView.DataSource = dt1;
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }

    private void idTextBox1_TextChanged(object sender, EventArgs e)
    {

    }

    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void k_st_cTextBox_TextChanged(object sender, EventArgs e)
    {

    }

    private void vubratuCheckBox_CheckedChanged(object sender, EventArgs e)
    {

    }

    private void button6_Click(object sender, EventArgs e)
    {
        try
        {

            dt2.Rows.Add( textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text);
            table2DataGridView.DataSource = dt2;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }

    private void button8_Click(object sender, EventArgs e)
    {
        try
        {

            dt3.Rows.Add(idTextBox3.Text, id_CilTextBox.Text, id_ZasibTextBox.Text,
imovirnist_1TextBox.Text);
            table3DataGridView.DataSource = dt3;
```

```
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
        }

private void fillByToolStripButton_Click(object sender, EventArgs e)
{
    try
    {
        this.table3TableAdapter.FillBy(this.genAlgDataSet.Table3);
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

private void fillBy1ToolStripButton_Click(object sender, EventArgs e)
{
    try
    {
        this.table2TableAdapter.FillBy1(this.genAlgDataSet.Table2);
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

private void fillBy1ToolStripButton1_Click(object sender, EventArgs e)
{
    try
    {
        this.table3TableAdapter.FillBy1(this.genAlgDataSet.Table3);
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

private void fillBy2ToolStripButton_Click(object sender, EventArgs e)
{
    try
    {
        this.table3TableAdapter.FillBy2(this.genAlgDataSet.Table3);
    }
    catch (System.Exception ex)
    {
```

```
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void fillBy3ToolStripButton_Click(object sender, EventArgs e)
    {
        try
        {
            this.table3TableAdapter.FillBy3(this.genAlgDataSet.Table3);
        }
        catch (System.Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void fillBy3ToolStrip_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
    {
    }

    private void fillToolStripButton_Click(object sender, EventArgs e)
    {
        try
        {
            this.table2TableAdapter.Fill(this.genAlgDataSet.Table2);
        }
        catch (System.Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void fillToolStripButton1_Click(object sender, EventArgs e)
    {
        try
        {
            this.table3TableAdapter.Fill(this.genAlgDataSet.Table3);
        }
        catch (System.Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void fillToolStripButton2_Click(object sender, EventArgs e)
    {
        try
        {
            this.table2TableAdapter.Fill(this.genAlgDataSet.Table2);
        }
        catch (System.Exception ex)
```

```
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void fillToolStripButton_Click_1(object sender, EventArgs e)
    {
        try
        {
            this.table2TableAdapter.Fill(this.genAlgDataSet.Table2);
        }
        catch (System.Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void fillToolStripButton1_Click_1(object sender, EventArgs e)
    {
        try
        {
            this.table3TableAdapter.Fill(this.genAlgDataSet.Table3);
        }
        catch (System.Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form2 f2 = new Form2();
        f2.Show();
    }
        }
}
```

Фрагмент коду імітаційного моделювання бою

```
            Connection conn = null;
                    try {
                            conn =
DriverManager.getConnection("jdbc:mysql://localhost/new_schema?user=root&password=root");
                    } catch (SQLException e1) {
                            // TODO Auto-generated catch block
                            e1.printStackTrace();
                    }               // password
            ResultSet rs;
                    try {
```

```
                                        rs = conn.createStatement().executeQuery("SELECT *
FROM army");
                        for (; rs.next(); ) {
                           for (int i = 0; i < 1; ++i) {
                              Object o = rs.getObject("name");    // Is SQL the first column is
indexed

                              TankInterface tank = null;
                              Integer t1=null,t2=null;
                                    try
                                    {
                                          t1=Integer.valueOf(rs.getObject("x").toString());
                                          t2=Integer.valueOf(rs.getObject("y").toString());
                                          t1 = t1==null ? 1:t1;
                                          t2 = t2==null ? 1:t2;
                                          t1 = (int)(((double)t1 / (double)getWidth()) *
grtScreenWidth());
                                          t2 = (int)(((double)t2 / (double)getHeight()) *
grtScreenHeight());
                                          tank =
tankFactory.createTank(rs.getObject("name").toString(),t1,t2, true,
rs.getObject("model").toString());



                                          t1=Integer.valueOf(rs.getObject("x1").toString());
                                          t2=Integer.valueOf(rs.getObject("y1").toString());
                                          t1 = t1==null ? 1:t1;
                                          t2 = t2==null ? 1:t2;
                                          t1 = (int)(((double)t1 / (double)getWidth()) *
grtScreenWidth());
                                          t2 = (int)(((double)t2 / (double)getHeight()) *
grtScreenHeight());
                                          tank.addWayInterval(new WayIntervalModel(t1,
t2));
                                    }
                                    catch(NumberFormatException e){}

                                    bestCells=null;
                                    tankList.add(tank);
                              System.out.print(o.toString() + " ");
                           }

                           System.out.println(" ");
                        }
                           } catch (SQLException e1) {
                                    // TODO Auto-generated catch block
                                    e1.printStackTrace();
                           }

                           try {
```

```
            conn.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
jdbc:mysql://localhost/new_schema?user=root&password=root
untitled11, то є GRAILS проект.
driverClassName = "com.mysql.jdbc.Driver"
   username = "root"
   password = "root"

Шлях до БД
   development {
      dataSource {
         dbCreate = "update" // one of 'create', 'create-drop','update'
         url = "jdbc:mysql://localhost:3306/new_schema"
      }
   }
   test {
      dataSource {
         dbCreate = "update"
         url = "jdbc:mysql://localhost:3306/new_schema"
      }
   }
   production {
      dataSource {
         dbCreate = "update"
         url = "jdbc:mysql://localhost:3306/new_schema"
      }
   }
```

Фрагмент коду застосунку «Adjustment»

GridActivity

```
package com.example.f4st.adjustment;

import android.content.Intent;
import android.content.res.Configuration;
import android.graphics.Point;
import android.graphics.PointF;
import android.graphics.RectF;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.DialogFragment;
import android.support.v4.app.FragmentActivity;
```

```
import android.view.Display;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;


import com.example.f4st.adjustment.DialogFragments.CountdownTimerDialogFragment;
import com.example.f4st.adjustment.DialogFragments.FireAnalyzeDialogFragment;
import com.example.f4st.adjustment.DialogFragments.ObserverInputDialogFragment;
import com.example.f4st.adjustment.DialogFragments.ReportDialogFragment;
import com.example.f4st.adjustment.DialogFragments.SetTimerDialogFragment;
import com.example.f4st.adjustment.Mathematic.LinearLine;
import com.example.f4st.adjustment.Mathematic.MathUtil;
import com.example.f4st.adjustment.Plot.BoxModel;
import com.example.f4st.adjustment.Plot.CoordinateSystem;

import java.util.ArrayList;

public class GridActivity extends FragmentActivity implements
        SetTimerDialogFragment.SetTimerDialogFragmentListener,
        CountdownTimerDialogFragment.CountdownTimerDialogFragmentListener,
        ObserverInputDialogFragment.ObserverInputDialogFragmentListener,
        ReportDialogFragment.ReportDialogFragmentListener,
        FireAnalyzeDialogFragment.FireAnalyzeDialogFragmentListener
        /*View.OnClickListener*/ {


    /**
     * Constant based on horizontal number of cell
     * 13 x 9 | 13 x 9
     * _______|_______
     * |
     * 13 x 9 | 13 x 9
     */
    private static final int MINIMUM_NUMBER_OF_CELLS = 26;

    private static final String TIMER_START_VALUE = "timer_start_value";
    private static final String CLOSEST_POINT_INDEX = "closest_point_index";
    private static final String LAST_POINT_INDEX = "last_point_index";
    private static final String ADJUSTMENT_TEXT = "adjustment_text";
    private static final String ADJUSTMENT_VISIBILITY = "adjustment_visibility";
    private static final String IS_LESS_SELECTED = "is_less_selected";
    private static final String IS_200_SELECTED = "is_200_selected";
    private static final String LIST_OF_POINTS = "list_of_points";
```

```
    private CanvasView mCanvasView;
    private CanvasModel mCanvasModel;
    private TextView mClosestPointTextView;
    private TextView mAdjustmentValueTextView;
    private Button mFlightTimerButton;
    private long mTimerStartValue;
    private int mClosestPointIndex;
    private int mLastPointIndex;
    private String mAdjustmentText;
    private boolean mAdjustmentVisibility;

    private DialogManager dialogManager;
    private CommandCreator commandCreator;

    private boolean mIsLessSelected;
    private boolean mIs200Selected;
    private ArrayList<HitPoint> mListOfPoints;

public interface ReportDialogFragmentListener {

    void onBackPressedInDialog(ReportDialogFragment dialog);
}

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.grid_activity);

        if (savedInstanceState != null) {
            restoreFromState(savedInstanceState);
        } else {
            mCanvasModel = new CanvasModel();
            mListOfPoints = new ArrayList<>();
            mClosestPointIndex = 0;
            mLastPointIndex = 0;
        }

        initPointTextView();
        initAdjustmentValueTextView();
        initFlightTimerButton();
        initFireButton();
        initPochatokButton();
              initManagers();

        mFlightTimerButton.setText(getString(R.string.flight_time) + ": " + mTimerStartValue /
1000);

        DeviceScreenMetrics deviceScreenMetrics = new DeviceScreenMetrics();
        int width = deviceScreenMetrics.getMeasuredWidth();
        int height = deviceScreenMetrics.getMeasuredHeight();
```

```
        mCanvasModel.setMinNumberOfCells(MINIMUM_NUMBER_OF_CELLS);
//        mCanvasModel.setBoxModel(new BoxModel(10, 10, 10, 10, 0, 0, 0, 0));
        mCanvasModel.setBoundsRect(new RectF(0, 0, width, height));
        CoordinateSystem.getInstance().setBoundsRect(new RectF(0, 0, width, height));

        mCanvasView = new CanvasView(this);

        mCanvasView.updateModel(mCanvasModel);

        updateClosestPointTextView();
        updateAdjustmentValueTextView();

        LinearLayout relativeLayout = (LinearLayout) findViewById(R.id.linear_layout);
        relativeLayout.addView(mCanvasView);


    }


    @Override
    protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        restoreFromState(savedInstanceState);
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        saveState(outState);
        super.onSaveInstanceState(outState);
    }

    public void restoreFromState(Bundle savedInstanceState) {
        mTimerStartValue = savedInstanceState.getLong(TIMER_START_VALUE);
        mClosestPointIndex = savedInstanceState.getInt(CLOSEST_POINT_INDEX);
        mLastPointIndex = savedInstanceState.getInt(LAST_POINT_INDEX);
        mAdjustmentText = savedInstanceState.getString(ADJUSTMENT_TEXT);
        mAdjustmentVisibility = savedInstanceState.getByte(ADJUSTMENT_VISIBILITY) != 0x00;
        mIsLessSelected = savedInstanceState.getByte(IS_LESS_SELECTED) != 0x00;
        mIs200Selected = savedInstanceState.getByte(IS_200_SELECTED) != 0x00;
        mListOfPoints = savedInstanceState.getParcelableArrayList(LIST_OF_POINTS);
        mCanvasModel = new CanvasModel(savedInstanceState);
    }

    public void saveState(Bundle outState) {
        outState.putLong(TIMER_START_VALUE, mTimerStartValue);
        outState.putInt(CLOSEST_POINT_INDEX, mClosestPointIndex);
        outState.putInt(LAST_POINT_INDEX, mLastPointIndex);
        outState.putString(ADJUSTMENT_TEXT, mAdjustmentText);
        outState.putByte(ADJUSTMENT_VISIBILITY, (byte) (mAdjustmentVisibility ? 0x01 :
```

```
0x00));
        outState.putByte(IS_LESS_SELECTED, (byte) (mIsLessSelected ? 0x01 : 0x00));
        outState.putByte(IS_200_SELECTED, (byte) (mIs200Selected ? 0x01 : 0x00));
        outState.putParcelableArrayList(LIST_OF_POINTS, mListOfPoints);
        mCanvasModel.saveInstanceState(outState);
        outState.putParcelable(CanvasModel.CANVAS_MODEL, mCanvasModel);
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
    }

    private void initPointTextView() {
        mClosestPointTextView = (TextView) findViewById(R.id.point_textView);
    }

    private void initAdjustmentValueTextView() {
        mAdjustmentValueTextView = (TextView) findViewById(R.id.adjustment_value_textView);
    }

    private void updateClosestPointTextView() {
        String text = getText(R.string.closest) + "\n"
                + getText(R.string.point).toString().toLowerCase()
                + ": ";
        if (mListOfPoints != null) {
            if (mListOfPoints.size() == 0) {
                text = text + 0;
            } else {
                text = text + (mClosestPointIndex + 1);
            }
            mClosestPointTextView.setText(text);
        }
    }

    private void updateAdjustmentValueTextView() {
        if (mAdjustmentVisibility) {
            mAdjustmentValueTextView.setVisibility(View.VISIBLE);
            mAdjustmentValueTextView.setText(mAdjustmentText);
        }
    }

    private void initFlightTimerButton() {
        mFlightTimerButton = (Button) findViewById(R.id.flight_timer_button);
        mFlightTimerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                dialogManager.showSetTimerDialog();
            }
        });
    }
```

```
private void initManagers() {
   dialogManager = new DialogManager(getSupportFragmentManager());
   commandCreator = new CommandCreator(getResources());
}

private void initFireButton() {
   final Button adjustmentFireButton = (Button) findViewById(R.id.fire_button);
   adjustmentFireButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(final View view) {

         dialogManager.showTimerDialog(mTimerStartValue);
      }
   });
}

@Override
public void onTimerSet(SetTimerDialogFragment dialog, int timerValue) {
   dialog.dismiss();
   mFlightTimerButton.setText(getString(R.string.flight_time) + ": " + timerValue);
   mTimerStartValue = timerValue * 1000;

}

@Override
public void onTimerExpireInDialog(CountdownTimerDialogFragment dialog) {
   dialog.dismiss();
   dialogManager.showObserverInputDialog();
}

@Override
public void onObserverInputSetInDialog(ObserverInputDialogFragment dialog,
                                       boolean isLeftSelected, int x,
                                       boolean isUpSelected, int y) {

   dialog.dismiss();

   float theX = (isLeftSelected) ? -x : x;
   float theY = (isUpSelected) ? y : -y;

   int index = mListOfPoints.size();
   mListOfPoints.add(index, new HitPoint(theX, theY, index, mClosestPointIndex, true));

   mLastPointIndex = mCanvasModel.setPoint(theX, theY, mClosestPointIndex);
   mCanvasView.updateModel(mCanvasModel);


   if (mLastPointIndex == 1) {
      ArrayList<LinearLine> lines = mCanvasModel.getListOfLines();
      LinearLine firstLine = lines.get(0);
```

```
            updateLineBasedOn(firstLine, mListOfPoints, mLastPointIndex);
        } else if (mLastPointIndex == 2) {
            ArrayList<LinearLine> lines = mCanvasModel.getListOfLines();
            LinearLine secondLine = lines.get(1);
            updateLineBasedOn(secondLine, mListOfPoints, mLastPointIndex);
        }

        if (index== 2) {
            setParallelLinesAndCalculateAdjustment();
        }

        if (mLastPointIndex == 0) {
            // show fire analyze aim dialog
            dialogManager.showFireAnalyzeDialog(mLastPointIndex);
        } else if (mLastPointIndex <= 2) {

            if (mLastPointIndex != mClosestPointIndex) {
                MathUtil mathUtil = new MathUtil();
                PointF lastPoint = mListOfPoints.get(mLastPointIndex).pointF();
                PointF closestPoint = mListOfPoints.get(mClosestPointIndex).pointF();
                PointF reallyClosest = mathUtil.closestPoint(closestPoint, lastPoint);
                if (reallyClosest.equals(lastPoint.x, lastPoint.y)) {
                    mClosestPointIndex = mLastPointIndex;
                }
            }
            // draw the line and show analyze point dialog
            if (mLastPointIndex < 2) {
                dialogManager.showFireAnalyzeDialog(mLastPointIndex);
            }
        }
        updateClosestPointTextView();
    }

    private LinearLine updateLineBasedOn(LinearLine line,
                                         ArrayList<HitPoint> listOfPoints,
                                         int lastPointIndex) {
        HitPoint lastPoint = listOfPoints.get(lastPointIndex);
        int previousPointIndex = lastPoint.getClosestPointIndex();
        HitPoint previousPoint = listOfPoints.get(previousPointIndex);
        int dividersCount = previousPoint.getLineDividersCount();
        line.updateLineData(previousPoint.pointF(), lastPoint.pointF(), dividersCount);
        return line;
    }

    private void setParallelLinesAndCalculateAdjustment() {
        ArrayList<PointF> listOfParallelLinePoints = new ArrayList<>();

        MathUtil mathUtil = new MathUtil();

        HitPoint firstPoint = mListOfPoints.get(1);
        HitPoint firstClosestPoint = mListOfPoints.get(0);
```

```
        ArrayList<PointF> parallelLinePoints = mathUtil.pointCoordinatesForParallelLine
                (firstPoint.pointF(),
                        firstClosestPoint.pointF());

        listOfParallelLinePoints.add(0, parallelLinePoints.get(0));
        listOfParallelLinePoints.add(1, parallelLinePoints.get(1));

        HitPoint secondPoint = mListOfPoints.get(2);
        int secondPointClosestPointIndex = secondPoint.getClosestPointIndex();
        HitPoint secondClosestPoint = mListOfPoints.get(secondPointClosestPointIndex);

        parallelLinePoints = mathUtil.pointCoordinatesForParallelLine(secondPoint.pointF(),
                secondClosestPoint.pointF());

        listOfParallelLinePoints.add(2, parallelLinePoints.get(0));
        listOfParallelLinePoints.add(3, parallelLinePoints.get(1));




        mCanvasModel.setListOfParallelLinePoints(listOfParallelLinePoints);
        mCanvasView.updateModel(mCanvasModel);

        calculateAdjustment(mCanvasModel);
    }

    private void calculateAdjustment(CanvasModel canvasModel) {
        ArrayList<LinearLine> lines = canvasModel.getListOfLines();
        LinearLine firstLine = lines.get(0);
        LinearLine secondLine = lines.get(1);
        MathUtil mathUtil = new MathUtil();

        float lessMoreAdjustment = mathUtil.findAdjustmentValue(
                firstLine.firstPoint,
                firstLine.secondPoint,
                secondLine.firstPoint,
                secondLine.secondPoint,
                true, firstLine.aimValue);

        float leftRightAdjustment = mathUtil.findAdjustmentValue(
                firstLine.firstPoint,
                firstLine.secondPoint,
                secondLine.firstPoint,
                secondLine.secondPoint,
                false, secondLine.rotationValue);

        // TODO: adjustment should be updated according to next request"
if (mLastPointIndex == 2) {
            mAdjustmentText = String.format("Adjustment based on \n %d point:" +
                    "\n lessMoreAdjustment = %6.1f \n, leftRightAdjustment = %.2f",
```

```
            (mClosestPointIndex + 1), lessMoreAdjustment, leftRightAdjustment);

        mAdjustmentVisibility = true;
        updateAdjustmentValueTextView();

    } else {
        if (mLastPointIndex == 3) {

            mAdjustmentText = String.format("Adjustment based on \n %d point:" +
                    "\n lessMoreAdjustment = %6.0f \n, leftRightAdjustment = %.2f",
                (mClosestPointIndex + 2), lessMoreAdjustment, leftRightAdjustment);

            mAdjustmentVisibility = true;
            updateAdjustmentValueTextView();
        }

@Override
public void onNextPressedInDialog(ReportDialogFragment dialog) {
    dialog.dismiss();

}

@Override
public void onBackPressedInDialog(ReportDialogFragment dialog) {
    dialog.dismiss();
}

@Override
public void onFireAnalyzeSetInDialog(FireAnalyzeDialogFragment dialog,
                                     boolean isLessSelected, boolean is200Selected,
                                     boolean isRightSelected, boolean is20Selected) {

    dialog.dismiss();

    // line dividers count depends only from is200Selected value. Sets for each line.

    mIsLessSelected = isLessSelected;
    mIs200Selected = is200Selected;
    int lineDividersCount = (mIs200Selected) ? 4 : 8;

    mCanvasModel.setLineDividersCount(lineDividersCount);
    mCanvasView.updateModel(mCanvasModel);

    String text, value, command;
    String basePointText = " " + (mClosestPointIndex+1) + " ";

    // show this dialog only for first  shot
    if (mLastPointIndex == 0) {

        if (mIsLessSelected) {
            text = getResources().getString(R.string.less);
```

```
            } else {
               text = getResources().getString(R.string.more);
            }

            if (mIs200Selected) {
               value = "200";
            } else {
               value = "400";
            }

            LinearLine firstLine = new LinearLine();
            firstLine.aimValue = mIs200Selected ? 200 : 400;
            mCanvasModel.setLine(firstLine);
            command = commandCreator.makeCommandFromAimData(text, value, basePointText);
         } else {

            if (isRightSelected) {
               text = getResources().getString(R.string.right);
            } else {
               text = getResources().getString(R.string.left);
            }

            if (is20Selected) {
               value = "0-20";
            } else {
               value = "0-40";
            }

            if (mLastPointIndex == 1) {
               LinearLine secondLine = new LinearLine();
               secondLine.aimValue = mIs200Selected ? 200 : 400;
               secondLine.rotationValue = is20Selected ? 20 : 40;
               mCanvasModel.setLine(secondLine);
            }
            command = commandCreator.makeCommandFromRotationData(text, value,
basePointText);
         }
         dialogManager.showReportDialog(command);
      }


      public class DeviceScreenMetrics
      {
         private int mMeasuredWidth;
         private int mMeasuredHeight;

         DeviceScreenMetrics() {
            WindowManager w = getWindowManager();
            final Point point = getDisplaySize(w.getDefaultDisplay());
            mMeasuredWidth = point.x;
            mMeasuredHeight = point.y;
```

```
        }

        @SuppressWarnings("deprecation") // deprecation related to methods on older devices
        private Point getDisplaySize(final Display display) {
            final Point point = new Point();
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
                display.getSize(point);
            } else { // Older device
                point.x = display.getWidth();
                point.y = display.getHeight();
            }
            return point;
        }

        public int getMeasuredWidth() {
            return mMeasuredWidth;
        }

        public int getMeasuredHeight() {
            return mMeasuredHeight;
        }
    }

private void initPochatokButton() {
        final Button pochatokButton = (Button) findViewById(R.id.pochatok_button);
        pochatokButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                onBackPressed1();
            }
        });

    }

    private void onBackPressed1() {
        super.onBackPressed();
    }
}
```

DrawManeger

```
package com.example.f4st.adjustment;

import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.PointF;
import android.graphics.RectF;

import com.example.f4st.adjustment.Mathematic.LinearLineFormula;
import com.example.f4st.adjustment.Mathematic.MathUtil;
import com.example.f4st.adjustment.Plot.CoordinateSystem;
```

```
import java.util.ArrayList;


public class DrawManager {

   private static DrawManager mInstance = null;

   private ArrayList<DrawMarker> drawMarkerList;
   private int squareSideSize;
   private CanvasModel mCanvasModel;



   private DrawManager(){
      // Empty constructor
   }

   public static DrawManager getInstance(){
      if(mInstance == null)
      {
         mInstance = new DrawManager();
      }
      return mInstance;
   }

   public void setCanvasModel(CanvasModel canvasModel)
   {
      this.mCanvasModel = canvasModel;
   }

   /**
    * ===== DRAW FULL VIEW =====
    */

   public void redraw(Canvas canvas, Paint mBackgroundPaint, Paint mAxisPaint, Paint
mGridPaint,
                  Paint mTextPaint, Paint mTextBackgroundPaint, Paint mPointPaint){

      CoordinateSystem coordinateSystem = CoordinateSystem.getInstance();

      // 1. draw canvas
      canvas.drawPaint(mBackgroundPaint);

      // 2. draw grid
      drawGridWithAxisOnCanvas(canvas, mCanvasModel.getBoundsRect(), mAxisPaint,
mGridPaint,
            mTextPaint, mTextBackgroundPaint);

      // 3. draw parallel lines
      if (mCanvasModel.wasParallelLineListSet()) {
```

```
        drawLineParallelLines(canvas, mCanvasModel.getListOfParallelLinePoints(),
mPointPaint);
    }

    ArrayList<HitPoint> listOfPoints = mCanvasModel.getListOfPoints();


    for(int i=0; i < listOfPoints.size(); i++) {

        HitPoint point = listOfPoints.get(i);
        HitPoint closestPoint = listOfPoints.get(point.getClosestPointIndex());

        HitPoint androidPoint = coordinateSystem.androidPoint(point);
        HitPoint androidClosestPoint = coordinateSystem.androidPoint(closestPoint);

        // 4. draw lines
        if (listOfPoints.size() > 1 && !point.hasEqualCoords(closestPoint)) {
            drawLineBetweenPoints(canvas, androidPoint, androidClosestPoint, mAxisPaint);
        }
    }

    for(int i=0; i < listOfPoints.size(); i++) {

        HitPoint point = listOfPoints.get(i);
        HitPoint androidPoint = coordinateSystem.androidPoint(point);

        // 5. draw point with numbers
        drawPointWithNumber(canvas, ""+ (i+1), androidPoint.getX(),
                androidPoint.getY(), mPointPaint, mTextPaint);
        if (listOfPoints.size()==5) {

            System.exit(0);

        }

    }
}


/**
 * ===== DRAW Methods =====
 */
private void drawLineParallelLines(Canvas canvas, ArrayList<PointF> listOfParallelLinePoints,
                                   Paint lineColor){

    CoordinateSystem coordinateSystem = CoordinateSystem.getInstance();

    final PointF A1 = listOfParallelLinePoints.get(0);
    final PointF A2 = listOfParallelLinePoints.get(1);

    final PointF B1 = listOfParallelLinePoints.get(2);
```

```
    final PointF B2 = listOfParallelLinePoints.get(3);

    ArrayList<PointF> resultLineACoordinates = new ArrayList<>();
    resultLineACoordinates.add(0, new PointF(0, 0));
    resultLineACoordinates.add(1, new PointF(0, 0));
    makeLineLonger(resultLineACoordinates, A1, A2);

    ArrayList<PointF> resultLineBCoordinates = new ArrayList<>();
    resultLineBCoordinates.add(0, new PointF(0, 0));
    resultLineBCoordinates.add(1, new PointF(0, 0));
    makeLineLonger(resultLineBCoordinates, B1, B2);

    PointF modifiedA1 = resultLineACoordinates.get(0);
    PointF modifiedA2 = resultLineACoordinates.get(1);

    PointF modifiedB1 = resultLineBCoordinates.get(0);
    PointF modifiedB2 = resultLineBCoordinates.get(1);

    drawLineBetweenPoints(canvas, coordinateSystem.mapX(modifiedA1.x),
            coordinateSystem.mapY(modifiedA1.y),
            coordinateSystem.mapX(modifiedA2.x),
            coordinateSystem.mapY(modifiedA2.y), 0, lineColor);

    drawLineBetweenPoints(canvas, coordinateSystem.mapX(modifiedB1.x),
            coordinateSystem.mapY(modifiedB1.y),
            coordinateSystem.mapX(modifiedB2.x),
            coordinateSystem.mapY(modifiedB2.y), 0, lineColor);
}

/**
 * Return coordinates to draw longer line
 * @param resultLineCoordinates - the return coordinates
 * @param firstPoint - previous line start
 * @param secondPoint - previous line end
 */
private void makeLineLonger(ArrayList<PointF> resultLineCoordinates,
                            PointF firstPoint,
                            PointF secondPoint) {


    int MAX_Y = 10;
    int MIN_Y = -10;
    int MAX_X = 40;
    int MIN_X = -40;

    PointF A1 = resultLineCoordinates.get(0);
    PointF A2 = resultLineCoordinates.get(1);

    MathUtil mathUtil = new MathUtil();

    // making line longer with a help of equation
```

```
        LinearLineFormula firstLineFormula = new LinearLineFormula();
        firstLineFormula.mSlope = mathUtil.slope(firstPoint, secondPoint);

        if (firstLineFormula.mSlope == 0.0) {
            // in case lines are on has the same point x OR y
            if (firstPoint.equals(secondPoint.x, secondPoint.y)) {
                A1 = firstPoint;
                A2 = secondPoint;
            } else if (firstPoint.x == secondPoint.x) {
                A1 = new PointF(firstPoint.x, MAX_Y);
                A2 = new PointF(firstPoint.x, MIN_Y);
            } else { // firstPoint.y == secondPoint.y
                A1 = new PointF(MIN_X, firstPoint.y);
                A2 = new PointF(MAX_X, firstPoint.y);
            }
        } else {
            firstLineFormula.mIntercept = mathUtil.intercept(firstPoint, firstLineFormula.mSlope);

            A1.x =
firstLineFormula.calculateX(firstLineFormula.mSlope,firstLineFormula.mIntercept,MAX_Y);
            A2.x =
firstLineFormula.calculateX(firstLineFormula.mSlope,firstLineFormula.mIntercept,MIN_Y);

            A1.y =
firstLineFormula.calculateY(firstLineFormula.mSlope,firstLineFormula.mIntercept,A1.x);
            A2.y =
firstLineFormula.calculateY(firstLineFormula.mSlope,firstLineFormula.mIntercept,A2.x);

            if (A1.x > MAX_X) {
                A1.x = MAX_X;
                A2.x = MIN_X;
                A1.y =
firstLineFormula.calculateY(firstLineFormula.mSlope,firstLineFormula.mIntercept,A1.x);
                A2.y =
firstLineFormula.calculateY(firstLineFormula.mSlope,firstLineFormula.mIntercept,A2.x);
            }
        }

        resultLineCoordinates.set(0,A1);
        resultLineCoordinates.set(1,A2);
    }

    private void drawGridWithAxisOnCanvas(Canvas canvas, RectF boundsRect, Paint axisPaint,
                                          Paint gridPaint, Paint markerTextPaint,
                                          Paint markerBackgroundPaint) {

//        mBoundsRect = boundsRect;
        int width = (int)boundsRect.width();
        int height = (int)boundsRect.height();

        // draw axis lines
```

```
float centerPositionY = (height - axisPaint.getStrokeWidth())/2;
float centerPositionX = (width - axisPaint.getStrokeWidth())/2;
canvas.drawLine(boundsRect.left, centerPositionY, width, centerPositionY, axisPaint);
canvas.drawLine(centerPositionX, boundsRect.top, centerPositionX, height, axisPaint);

// find minimum side and calculate square size.
int min = Math.min(width , height);
squareSideSize = (min / mCanvasModel.getNumberOfCells());

CoordinateSystem.getInstance().setSquareSideSize(squareSideSize);

// list to store draw markers and draw then at the end (to avoid line overriding)
drawMarkerList = new ArrayList<>();

// counter for adding markers to grid
int counter = 0;

// draw lines from center to lower X
for (int i = width/2 - squareSideSize; i >= boundsRect.left; i -=squareSideSize) {
   counter++;
   canvas.drawLine(i, boundsRect.top, i, height, gridPaint);
   // each 2 cells should be markers
   if (counter%2 == 0) {
      int pointValue = counter * 5;
      int index = drawMarkerList.size();
      drawMarkerList.add(index, new DrawMarker(""+ pointValue, i, boundsRect.centerY()));
   }
}

// draw lines from center to higher X
counter = 0;
for (int i = width/2 + squareSideSize; i <= boundsRect.right; i +=squareSideSize) {
   counter++;
   canvas.drawLine(i, boundsRect.top, i, height, gridPaint);
   // each 2 cells should be markers
   if (counter%2 == 0) {
      int pointValue = counter * 5;
      int index = drawMarkerList.size();
      drawMarkerList.add(index, new DrawMarker(""+ pointValue, i, boundsRect.centerY()));
   }
}

// draw lines from center to lower Y
counter = 0;
for (int i = height/2 + squareSideSize; i <= boundsRect.bottom; i +=squareSideSize) {
   counter++;
   canvas.drawLine(boundsRect.left, i, width, i, gridPaint);
   // each 5 cells should be markers
   if (counter%5 == 0) {
      int index = drawMarkerList.size();
      drawMarkerList.add(index, new DrawMarker(""+ counter, boundsRect.centerX(), i));
```

```
            }
        }

        // draw lines from center to higher Y
        counter = 0;
        for (int i = height/2 - squareSideSize; i >= boundsRect.top; i -=squareSideSize) {
            counter++;
            canvas.drawLine(boundsRect.left, i, width, i, gridPaint);
            // each 5 cells should be markers
            if (counter%5 == 0) {
                int index = drawMarkerList.size();
                drawMarkerList.add(index, new DrawMarker(""+ counter, boundsRect.centerX(), i));
            }
        }


        drawMarkers(canvas, markerTextPaint, markerBackgroundPaint);
        drawMarkerList.clear();
    }

    private void drawMarkers(Canvas canvas, Paint textPaint, Paint backgroundPaint) {
        for (DrawMarker drawMarker : drawMarkerList) {
            drawTextWithBackground(canvas, drawMarker.getText(),
                    drawMarker.getX(), drawMarker.getY(), textPaint, backgroundPaint);
        }
    }

    private void drawTextWithBackground(Canvas canvas, String text, float x, float y,
                                        Paint textPaint, Paint backgroundPaint) {

        int margin = 2;
        Paint.FontMetrics fontMetrics = new Paint.FontMetrics();
        textPaint.getFontMetrics(fontMetrics);

        float textHeight = fontMetrics.bottom - fontMetrics.top;
        float textWidth = textPaint.measureText(text);

        float startX = x - textWidth/2;
        float startY = y + textHeight/2;

        canvas.drawRect(startX - margin, startY + fontMetrics.top - margin, startX + textWidth +
margin,
                startY + fontMetrics.bottom + margin, backgroundPaint);
        canvas.drawText(text, startX, startY, textPaint);
    }

    private void drawPointWithNumber(Canvas canvas, String number, float x, float y,
                                     Paint pointPaint, Paint textPaint) {
        int margin = 2;
        Paint.FontMetrics fontMetrics = new Paint.FontMetrics();
        textPaint.getFontMetrics(fontMetrics);
```

```
    float textHeight = fontMetrics.bottom - fontMetrics.top;
    float textWidth = textPaint.measureText(number);

    float startX = x - textWidth/2;
    float startY = y + textHeight/2;

    float radius = Math.max(textHeight,textWidth)/2;
    canvas.drawCircle(x, y, radius, pointPaint);
    canvas.drawText(number,startX,startY-margin,textPaint);
}

private void drawLineBetweenPoints(Canvas canvas, HitPoint firstPoint, HitPoint secondPoint,
                                   Paint linePaint) {

    drawLineBetweenPoints(canvas, firstPoint.getX(), firstPoint.getY(),
            secondPoint.getX(), secondPoint.getY(),
            secondPoint.getLineDividersCount(), linePaint);
}

private void drawLineBetweenPoints(Canvas canvas, float x1, float y1, float x2, float y2,
                                   int lineDividersCount, Paint paint) {
    CoordinateSystem coordinateSystem = CoordinateSystem.getInstance();

    canvas.drawLine(x1, y1, x2, y2, paint);

    if (lineDividersCount > 1) {

        // midPoint is in cartesian coordinate system, while x1, y1, x2, y2 in android
        PointF lineMidPoint = drawMidPoint(canvas, x1, y1, x2, y2, paint);
        lineDividersCount = lineDividersCount/2;
        if (lineDividersCount > 1) {

            // pass android coordinates here
            drawLineBetweenPoints(canvas, x1, y1, coordinateSystem.mapX(lineMidPoint.x),
                    coordinateSystem.mapY(lineMidPoint.y),
                    lineDividersCount, paint);
            drawLineBetweenPoints(canvas, coordinateSystem.mapX(lineMidPoint.x),
                    coordinateSystem.mapY(lineMidPoint.y), x2, y2,
                    lineDividersCount, paint);
        }
    }
}

private PointF drawMidPoint(Canvas canvas, float x1, float y1, float x2, float y2, Paint paint) {
    CoordinateSystem coordinateSystem = CoordinateSystem.getInstance();

    float distanceFromMidPointCenterToEnd = 2;

    HitPoint firstPoint = coordinateSystem.cartesianPoint(new HitPoint(x1, y1, 0, 0, false));
    HitPoint secondPoint = coordinateSystem.cartesianPoint(new HitPoint(x2, y2, 0, 0, false));
```

```
        MathUtil mathUtil = new MathUtil();
        PointF lineMidPoint = mathUtil.findMidpoint(firstPoint.getX(), firstPoint.getY(),
                secondPoint.getX(), secondPoint.getY());

        float slope = mathUtil.slope(firstPoint.pointF(), secondPoint.pointF());

        PointF lineMidPointTop;
        PointF lineMidPointBottom;

        if (slope == 0.0) {
           // in case lines are on has the same point x OR y

           if (firstPoint.hasEqualCoords(secondPoint)) {
              lineMidPointTop = new PointF(lineMidPoint.x, lineMidPoint.y);
              lineMidPointBottom = new PointF(lineMidPoint.x, lineMidPoint.y);
           } else if (firstPoint.getX() == secondPoint.getX()) {
              lineMidPointTop = new PointF(lineMidPoint.x + distanceFromMidPointCenterToEnd,
                    lineMidPoint.y);
              lineMidPointBottom = new PointF(lineMidPoint.x - distanceFromMidPointCenterToEnd,
                    lineMidPoint.y);
           } else { // firstPoint.getY() == secondPoint.getY()
              lineMidPointTop = new PointF(lineMidPoint.x,
                    lineMidPoint.y + distanceFromMidPointCenterToEnd/5);
              lineMidPointBottom = new PointF(lineMidPoint.x,
                    lineMidPoint.y - distanceFromMidPointCenterToEnd/5);
           }
        } else {
           LinearLineFormula linearLineFormula = new LinearLineFormula();
           linearLineFormula.mSlope = mathUtil.perpendicularLineSlope(slope);

           linearLineFormula.mIntercept = mathUtil.intercept(lineMidPoint,
linearLineFormula.mSlope);
           lineMidPointTop = mathUtil.findEndPoint(lineMidPoint.x, lineMidPoint.y,
                 linearLineFormula.mSlope, distanceFromMidPointCenterToEnd);
           lineMidPointBottom = mathUtil.findEndPoint(lineMidPoint.x, lineMidPoint.y,
                 linearLineFormula.mSlope, - distanceFromMidPointCenterToEnd);

           float deltaY = lineMidPointTop.y - lineMidPoint.y;
           lineMidPointTop.y = lineMidPoint.y + deltaY/5;

           float deltaX = lineMidPointTop.x - lineMidPoint.x;
           lineMidPointTop.x = lineMidPoint.x + deltaX*5;

           deltaY = lineMidPointBottom.y - lineMidPoint.y;
           lineMidPointBottom.y = lineMidPoint.y + deltaY/5;

           deltaX = lineMidPointBottom.x - lineMidPoint.x;
           lineMidPointBottom.x = lineMidPoint.x + deltaX*5;
        }
```

```
        drawLineBetweenPoints(canvas, coordinateSystem.mapX(lineMidPointTop.x),
                coordinateSystem.mapY(lineMidPointTop.y),
                coordinateSystem.mapX(lineMidPointBottom.x),
                coordinateSystem.mapY(lineMidPointBottom.y), 0, paint);
        return lineMidPoint;
    }


    private class DrawMarker {

        private String mText;
        private float mX;
        private float mY;

        public DrawMarker(String mText, float mX, float mY) {
            this.mText = mText;
            this.mX = mX;
            this.mY = mY;
        }

        public String getText() {
            return mText;
        }

        public float getX() {
            return mX;
        }

        public float getY() {
            return mY;
        }
    }
}
```

ДОДАТОК Б

Фрагмент онтології СВ ЗСУ

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/Ontology1350143052.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1350143052.owl">
  <owl:Ontology rdf:about="">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
   <rdfs:domain>
    <owl:Class rdf:ID="Communication_system"/>
   </rdfs:domain>
  </owl:Ontology>
  <rdfs:Class rdf:ID="Military_equipment">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Військова техніка</rdfs:comment>
  </rdfs:Class>
  <owl:Class rdf:ID="Military_unit">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Connectivity"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Association_of"/>
   </rdfs:subClassOf>
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Військова_частина</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="Options">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Параметри</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="Rocket_Brigade">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Ракетні_бригади</rdfs:comment>
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Missile_Forces"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="_19_separate_missile_brigade">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >19 окрема ракетна бригада</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Directly_subordinate_command_of_the_army"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="BMP_2">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >65</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_maximum_speed"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="has_combat_weight"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >13.8</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >7</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_maximum_speed_afloat"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_ditch"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >2.5</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_ATRA"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Konkurs</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
<rdfs:subClassOf>
  <owl:Class rdf:ID="BM"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_crew"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >10</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_average_ground_pressure"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.63</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >bullets protection</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_armor"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_country_developer"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >USSR</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >6735</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_length_with_gun_forward"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
      >swimming</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_ford"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Fahot</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_ATGM"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <owl:Class rdf:ID="Charge"/>
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has_armor-piercing_projectile"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2000</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="has_ammunition_machine_gun"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >БМП-2</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >3150</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_width"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >R-123M
</owl:hasValue>
      <owl:onProperty>
```

```
      <owl:DatatypeProperty rdf:ID="has_radio_station"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >300</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_engine_power"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >30</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_caliber_gun"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >UTD-20</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_engine_type"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1980</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_adopting_year"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >7.62</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_caliber_machine_gun"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
```

```
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_range_view"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >2.6</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_model_machine_gun"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >PKT</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="has_amount_machine_guns"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_object"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >675</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >600</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_cruising_range"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >BPK-1-42</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="has_sight"/>
    </owl:onProperty>
  </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <owl:Class rdf:ID="_2A42"/>
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has_gun"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >8</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="has_installation_assault_rifle"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >21.8</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_power_density"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_lift"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >35</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2059</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="has_height"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="VCC-80">
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_length_with_gun_forward"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5430</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_engine_power"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >520</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3210</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_width"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_installation_assault_rifle"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >4</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >9</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_crew"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >500</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_cruising_range"/>
    </owl:onProperty>
  </owl:Restriction>
```

```
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_maximum_speed"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >70</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >25</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_caliber_gun"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#has_armor-piercing_projectile"/>
    </owl:onProperty>
    <owl:hasValue>
      <owl:Class rdf:about="#Charge"/>
    </owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_combat_weight"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >21.7</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_amount_machine_guns"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_ATGM"/>
```

```
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Tou</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has_height"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1920</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#has_country_developer"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Italy</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#BM"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >2.1</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has_range_view"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >VCC-80</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#has_gun"/>
      </owl:onProperty>
      <owl:hasValue>
        <owl:Class rdf:ID="_2A28"/>
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_8_separate_special_regiment">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Directly_subordinate_command_of_the_army"/>
```

```
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >8 окремий полк спеціального призначення</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Moving">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Рухомі</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Items_management"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Subdivisions">
  <rdfs:subClassOf rdf:resource="#Military_unit"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Підрозділ_забезпечення</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="_107_separate_reactive_artillery_regiment">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >107 окремий реактивний артилерійський полк</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_6_Army_corps"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_27_separate_reactive_artillery_regiment_">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >27 окремий реактивний артилерійський полк</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_8_Army_corps"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_55_separate_linear-nodal_communication_regiment">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_13_Army_corps"/>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >55 окремий лінойно-вузловий полк зв'язку</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Marder_A3">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#has_country_developer"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >FRG</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#has_armor-piercing_projectile"/>
```

```
    </owl:onProperty>
    <owl:hasValue>
      <owl:Class rdf:about="#Charge"/>
    </owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >20</owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_caliber_gun"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class rdf:about="#BM"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_maximum_speed_afloat"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >6</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_engine_power"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >600</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_width"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3001</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_combat_weight"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >33.5</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >2.5</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_range_view"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_cruising_range"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >600</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Milan-2</owl:hasValue>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_ATGM"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#has_crew"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >10</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>"Мардер" A3</rdfs:comment>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#has_maximum_speed"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >65</owl:hasValue>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
```

```
      <owl:Restriction>
       <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has_length_with_gun_forward"/>
       </owl:onProperty>
       <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >6430</owl:hasValue>
      </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >1987</owl:hasValue>
       <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has_adopting_year"/>
       </owl:onProperty>
      </owl:Restriction>
     </rdfs:subClassOf>
<owl:Restriction>
       <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >3001</owl:hasValue>
       <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has_width"/>
       </owl:onProperty>
      </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >6</owl:hasValue>
       <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has_maximum_speed_afloat"/>
       </owl:onProperty>
      </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >"Мардер" A3</rdfs:comment>
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#has_gun"/>
       </owl:onProperty>
       <owl:hasValue>
        <owl:Class rdf:about="#_2A28"/>
       </owl:hasValue>
      </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#has_crew"/>
       </owl:onProperty>
```

ДОДАТОК В

Акти про впровадження результатів дисертаційних досліджень

Державний концерн «УКРОБОРОНПРОМ»
ДЕРЖАВНЕ ПІДПРИЄМСТВО
ЛЬВІВСЬКИЙ НАУКОВО-ДОСЛІДНИЙ РАДІОТЕХНІЧНИЙ ІНСТИТУТ
(ДП ЛНДРТІ)

79060, Львів, вул. Наукова, 7, тел. / факс +38-032-229-77-79, телетайп 73-43-13 „Вольт”
Web сайт: http://lreri.tripod.com/index.html; email: lreri@ukr.net Код ЄДРПОУ 14311429

ЗАТВЕРДЖУЮ

Директор ДП ЛНДРТІ

І. І. Огородник

«04» 01 2016 р.

АКТ

щодо використання результатів дисертаційної роботи Оборської Оксани Володимирівни, представленої на здобуття вченого ступеня кандидата технічних наук

Комісія у складі:
голова комісії – доктор технічних наук, начальник відділення
Оліярник Богдан Олексійович,
члени комісії – кандидат технічних наук, начальник відділу
Євтушенко Костянтин Станіславович
начальник відділу
Смоков Сергій Петрович

встановила, що наукові положення, які розроблені О. В. Оборською у ході дисертаційного дослідження, використані під час виконання дослідницьких робіт із створення моделі планування вогню та прийняття рішень в перспективній автоматизованій системі управління тактичної ланки.

Зокрема, використано удосконалений метод цілерозподілу на основі методів штучного інтелекту, а саме генетичних алгоритмів та опрацювання онтологічних знань, який, на відміну від існуючих, не містить процедур повного перебору, що дало змогу зменшити обчислювальну складність алгоритму пошуку ефективного цілерозподілу.

Розроблені О. В. Оборською наукові положення дозволяють автоматизувати окремі дії, які виконує командир тактичної ланки управління у ході планування вогню.

Даний акт не є підставою для фінансових розрахунків.

Голова комісії
доктор технічних наук, начальник відділення Б. О. Оліярник

Члени комісії:
кандидат технічних наук, начальник відділу К. С. Євтушенко

начальник відділу С. П. Смоков

МІНІСТЕРСТВО ОБОРОНИ УКРАЇНИ
ВІЙСЬКОВА ЧАСТИНА
— А 2129 —
Код 07897921
„25" 12 2015р.
№ 1206
49006 м. Дніпропетровськ

ЗАТВЕРДЖУЮ
Командир військової частини А2129
_____ О.М.Ліщенко
«10» грудня 2015 року

АКТ

про реалізацію результатів дисертаційних досліджень аспіранта
Національного університету „Львівська політехніка"
Оборської Оксани Володимирівни

Комісія у складі: Дроздовського О О., членів комісії: Харченка С.В., Резуненко Н.В. встановила, що програмні реалізації наукових положень, розроблені Оборською О.В., являються ефективним інструментом навчання, який планується впровадити в систему бойової підготовки військової части А2129 у 2016 навчальному році.

До основних результатів наукових досліджень, що знайшли своє впровадження, слід віднести андроїд-додаток «Adjustment» у якому реалізований спосіб корегування вогню артилерії та передачі розвідувальних даних на основі онтологічного підходу, що на відміну від відомих способів дозволяє суттєво скоротити час на вогневе ураження цілі та отримати стійкі навички корегування вогню артилерії в процесі підготовки корегувальників артилерії, в тому числі позаштатних, котрі не мають фахових знань в питаннях артилерії.

Розроблені Оборською О.В. метод та програмні реалізації дають змогу автоматизувати процес корегування вогню артилерії, які виконують корегувальники вогню артилерії під час тренування і ведення бойових дій.

Голова комісії: О.О.ДРОЗДОВСЬКИЙ

Члени комісії: С.В.ХАРЧЕНКО

Н.В.РЕЗУНЕНКО

«ЗАТВЕРДЖУЮ»
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
_____ професор Н.І. Чухрай
«__» ____11_____ 2015 р.

А К Т

використання наукових результатів Оборської Оксани Володимирівни, отриманих у ході кандидатських дисертаційних досліджень за темою «Методи та засоби моделювання петлі Бойда у військових застосуваннях з використанням онтологічного підходу» у науково-дослідній роботі кафедри інформаційних систем та мереж

Комісія у складі. голови комісії начальника науково-дослідної частини, к.т.н., доцента Жук Л.В. та членів комісії завідувача кафедри інформаційних систем та мереж, д.т.н., професора Литвина В.В., завідувача відділу науково-організаційного супроводу наукових досліджень, к.т.н. Лазько Г.В. та заступника начальника планово-фінансового відділу Чулой Т.М., цим актом підтверджують, що результати дисертаційної роботи Оборської О.В. на тему «Методи та засоби моделювання петлі Бойда у військових застосуваннях з використанням онтологічного підходу» використано у науково-дослідній роботі «Розроблення інтелектуальних розподілених систем на основі онтологічного підходу з метою інтеграції інформаційних ресурсів» (№ держреєстрації 0115U004228), що виконувалась на кафедрі інформаційних систем та мереж.

Отримані автором результати використано:

під час розроблення математичних моделей, методів та засобів підтримки прийняття рішень у конкурентному середовищі з використанням онтологічного підходу;

- під час розроблення архітектури системи підтримки прийняття рішень, центральною компонентою якої є онтологія предметної області,
- під час побудови програмних модулів, що ґрунтуються на розроблених моделях та методах.

Голова комісії:
Начальник науково-дослідної частини
к.т.н., доцент Л.В. Жук

Члени комісії:
Завідувач кафедри інформаційних
систем та мереж, д.т.н., професор В.В. Литвин

Зав. відділу науково-організаційного
супроводу наукових досліджень, к.т.н. Г.В. Лазько

Заст начальника
планово-фінансового відділу Т.М. Чулой

«ЗАТВЕРДЖУЮ»
Проректор з науково-педагогічної роботи
Національного університету
«Львівська політехніка»
_____ О.Р. Давидчак
"16" 11 2015 р.

А К Т

про впровадження в навчальний процес результатів
кандидатської дисертаційної роботи
Оборської Оксани Володимирівни

Цей акт складено про те, що результати кандидатської дисертаційної роботи Оборської Оксани Володимирівни на тему «Методи та засоби моделювання петлі Бойда у військових застосуваннях з використанням онтологічного підходу», представленої на здобуття наукового ступеня кандидата технічних наук, використовуються у навчальному процесі кафедри «Інформаційні системи та мережі» Національного університету «Львівська політехніка». Матеріали дисертаційного дослідження використовуються під час написання студентами курсових робіт, кваліфікаційних бакалаврських та магістерських робіт, а також під час викладання дисциплін «Теорія прийняття рішень», «Технології підтримки процесів прийняття рішень».

Зокрема, у навчальному процесі використовуються запропоновані О.В.Оборською.

- процес підтримки прийняття рішень в конкурентному середовищі (дисципліна «Теорія прийняття рішень» для студентів освітньо-кваліфікаційного рівня «бакалавр», що навчаються за напрямом 6.050101 «Комп'ютерні науки», тема 5 «Система переваг особи, що приймає рішення», тема 6 «Узгодження систем переваг в задачах групового прийняття рішень»);

- технології прийняття рішень на основі онтологій (дисципліна «Технології підтримки процесів прийняття рішень» для студентів освітньо-кваліфікаційного рівня «магістр», що навчаються за напрямом 8.04030302 «Системи і методи прийняття рішень», тема 3 «Технологія сховищ даних в системах підтримки рішень», тема 4 «Перспективи розвитку систем підтримки прийняття рішень»).

Результати кандидатської дисертаційної роботи опубліковано у 19 наукових публікаціях. 6 опубліковано у фахових наукових виданнях, з них. 1 одноосібна, 1– закордонна; 13 – у матеріалах міжнародних конференцій.

Директор ІКНІ,
д.т.н., професор М.О Медиковський

Завідувач кафедри інформаційних
систем та мереж,
д.т.н., професор В.В.Литвин

Професор кафедри інформаційних
систем та мереж, д.т.н., професор Р.М. Камінський