Issue 13, 2023

https://doi.org/10.23939/sisn2023.13.300

УДК 004.9

ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА СИСТЕМА АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ УМОВНИХ ЗНАКІВ У СТАНДАРТІ APP-6D

Василь Литвин[1], Антон Ринков[1], Вікторія Висоцька[1, 2]

[1] Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж, Львів, Україна
[2] Університет Оснабрюка, Інститут комп'ютерних наук,
Оснабрюк, Німеччина
E-mail: Vasyl.V.Lytvyn@lpnu.ua, ORCID: 0000-0002-9676-0180
E-mail: Anton.Rynkov.mnsa.2019@lpnu.ua, ORCID: 0000-0002-5751-9052
E-mail: Victoria.A.Vysotska@lpnu.ua, ORCID: 0000-0001-6417-3689

Військова символіка відіграє ключову роль у командуванні військовими силами та їх контролі. Передаючи інформацію, що відповідає основним вимогам, можна швидко досягти ситуативного усвідомлення; за графічною сутністю вона забезпечує загальну операційну мову, що істотно полегшує взаємодію через культурні та мовні бар'єри. З появою інформаційних технологій виникла потреба у швидкому визнанні міжнародних стандартів, яким потім можна було б навчити комп'ютери. Злиття повітряних, морських та наземних символів із найвищим графічним стандартом на папері APP-6 привело до створення Mil-Std-2525 та НАТО APP-6D. Цей стандарт застосовують до всіх наземних компонентів НАТО, які прямо або частково беруть участь в операціях C4I, системних операціях, розробленні систем і навчанні в контексті операцій наземних компонентів НАТО. Під час проєктування та розроблення знаків, складання композиції їх складових необхідно ураховувати психологічні чинники, такі як здатність легко розпізнати знаки, їх розбірливість у різних умовах освітлення на різноманітній картографічній основі, на електронних екранах різних розмірів та типів за різних ступенів фізичної та інтелектуальної втоми. У Стандарті умовних знаків зібрано знаки для використання у сучасних мультихроматичних електронних системах. Водночас усі знаки можуть застосовуватися у монохроматичних системах та для нанесення обстановки від руки. Розроблення цього програмного забезпечення дасть змогу забезпечити комфортні умови для виконання таких дій, як створення і редагування тактичної графіки військової символіки; забезпечити дотримання усіх стандартів умов стандартизації APP-6D; виконання безпечного передавання символів між кількома користувачами. Об'єктом дослідження є автоматичний генератор умовних символів у тактичному військовому стандарті APP-6D. Предмет дослідження – програмне забезпечення для автоматичної генерації тактичних символів у стандарті APP-6D. Основна мета – розроблення програмного забезпечення, яке автоматично генеруватиме тактичні знаки у стандарті APP-6D. Виконано порівняльний аналіз різних методів розроблення таких систем генерації символів. Розроблене програмне забезпечення не має аналогів українською мовою, а також жодної десктопної реалізації. Проєкт може бути частиною програмного забезпечення для українських військових.

Ключові слова: автоматичний генератор; стандарт APP-6D; програмне забезпечення; графіка; символ; знак; інформаційна система.

Вступ. Постановка проблеми

Технології розвиваються швидкими темпами, і "програмне забезпечення як послуга" (SaaS) все більше стає стандартом для забезпечення функціональності. З цієї причини модель SaaS є одним з

основних трендів у сучасних вебтехнологіях. Часто SaaS розглядають як бізнес-модель, причому його часто помилково прирівнюють до оренди, що хоч і має такі самі загальні риси, але основна суть все ж в іншому: як правило, споживачів цікавлять технологічні особливостьі моделі, а не порядок і періодизація оплати.

Споживаючи ту чи іншу послугу як сервіс, а не як встановлене у себе програмне забезпечення, ми вже в цей момент, як правило, стаємо споживачами SaaS [4].

Пошта, за великим рахунком, це найпростіший приклад, і наймасовіший. Хоча б тому, що абсолютна більшість користувачів мережі електронною поштою користуються, а ось CRM, ERP системи або сайтбилдери потрібні далеко не всім. У них вужче коло споживачів. Нині дуже багато програмних продуктів вже можна знайти у вигляді сервісів. У деяких випадках вони, звичайно, ще відстають від своїх десктопних аналогів, але часто забезпечують своєму споживачеві переваги [4]. Здебільшого це виражається саме в простоті експлуатації, немає потреби в обслуговуванні, в економічній доцільності, інакше кажучи, в дешевизні такого рішення і підході.

Інші плюси залежать переважно вже від конкретної сфери, в деяких випадках це вищий ступінь безпеки, в інших безпрецедентно зручна синхронізація і доступність комфортної роботи багатьох користувачів, за рахунок хмарного зберігання даних. Частіше за інших, як приклади використання SaaS рішень, трапляються системи управління проєктами і спільної роботи над ними, онлайнові органайзери та системи документообігу. Не такими поширеними прикладами використання SaaS рішень також є всі наявні генератори умовних знаків. Вебсервіси надають функціональність для створення, збереження і обміну військовими символами і тактичною графікою.

Мета дослідження – розроблення програмного забезпечення, яке автоматично генеруватиме тактичні знаки у стандарті APP-6D. Об’єктом дослідження є автоматичний генератор умовних символів у тактичному військовому стандарті APP-6D. Предмет дослідження – програмне забезпечення для автоматичної генерації тактичних символів у стандарті APP-6D. Актуальність полягає у створенні десктопного рішення генератора. Десктопні додатки не потребують підключення до інтернету, їхня швидкодія вища. На момент виконання роботи не виявлено жодного подібного рішення. Це робить програмне забезпечення унікальним та дає можливість збільшити об’єм функціональності програми.

Аналіз останніх досліджень та публікацій

Для того, щоб з’ясувати рівень сучасних генераторів умовних знаків, виконано огляд та аналіз переваг і недоліків сучасних генераторів. Нижче наведемо короткий опис вебгенераторів і порівняльний аналіз їх можливостей.

ArcGIS – це географічна інформаційна система (ГІС) для роботи з картами та географічною інформацією. ГІС використовують для створення та використання карт, складання географічних даних, аналізу відображеної інформації, спільного використання та виявлення географічної інформації в деяких застосунках, а також для управління географічною інформацією в базі даних. Система забезпечує інфраструктуру для надання карт і географічної інформації у межах всієї організації, яка буде її використовувати. ArcGIS містить такі програми для Windows:

5. ArcReader, який дає змогу переглядати і запитувати карти, створені за допомогою інших продуктів ArcGIS;

6. ArcGIS for Desktop, який ліцензується на трьох рівнях функціональності:

· ArcGIS for Desktop Basic, який дає можливість переглядати просторові дані, створювати багатошарові карти і виконувати базовий просторовий аналіз;

· ArcGIS for Desktop Standard, який є застосунком до функціональності ArcView і містить просунутіші інструменти для маніпулювання шейп-файлами і базами геоданих;

· ArcGIS for Desktop Advanced, який надає можливості для опрацювання, редагування і аналізу даних.

Існують також серверні продукти ArcGIS, а також продукти ArcGIS для кишенькового персонального комп'ютера (КПК). Для підтримки військових операцій і планування військові використовують стандартні символи для візуалізації на полі битви. Ці символи представляють військові частини, обладнання та установки, а також тактичну графіку для позначення військових операцій, кордонів або інших спеціальних позначень. Символи також мають колірне кодування для позначення дружніх, ворожих або нейтральних об'єктів. Військовий редактор символів підтримує такі стандарти: MIL-STD-2525B; MIL-STD-2525D; APP-6 (D). Військові інструменти в ArcGIS – це набір програмних рішень, що спрощують процеси захисту та розвідки. Для того, щоб використовувати це програмне рішення, потрібно перейти на вкладку Military Tools for ArcGIS. Приклад роботи програми наведено на рис. 1.

Переваги генератора умовних знаків ArcGIS:

- Швидка конвертація між кількома стандартними форматами.
- Динамічне створення геодезичних ліній, кола, еліпса і кільця дальності.
- Можливість виконувати інтерактивний лінійний і радіальний аналіз прямої видимості.
- Можливість швидко і легко створювати стандартні військові символи.

Недоліки генератора:

- Підтримка тільки для операційної системи Microsoft Windows.
- Високі ціни на продукти.
- Власні формати даних.
- Труднощі перенесення даних із ГІС до інших програмних продуктів.

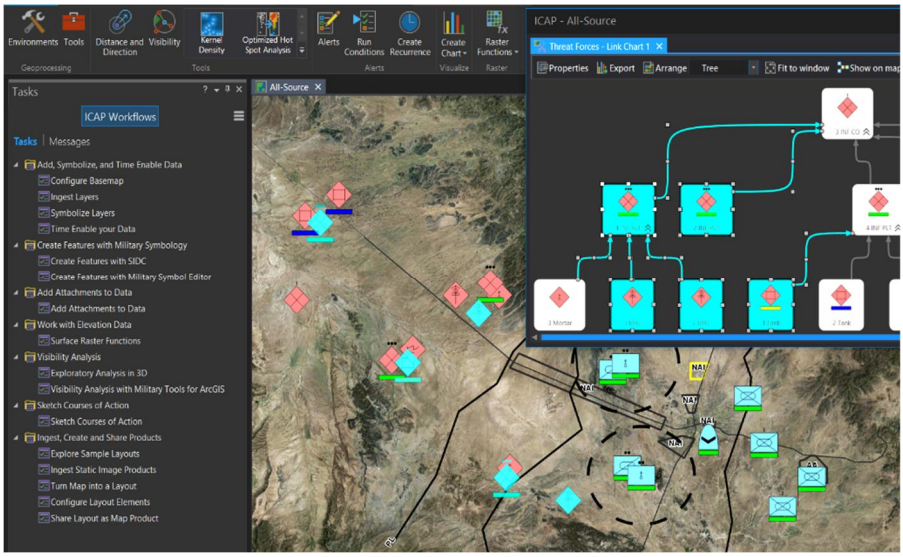


Рис. 1. Приклад роботи програми

Вебдодаток для генерування умовних знаків "Easy symbol", створений для створення тактичної графіки і редагування військових символів. Всю згенеровану графіку можна зберігати в звичайних векторних і графічних форматах або вона надається для загального доступу через URL [1]. "Easy symbol" звертається до вебслужби MSS (Military Symbol Service) для відтворення загальних символів бойової поведінки. Тобто застосунок дає змогу створювати, зберігати і поширювати всю загальну символіку бойових дій, зокрема тактичну графіку (*n*-точкові символи).

Генератор розміщений на сайті www.symbol.army. Концепція вебзастосунку надає перевагу, що полягає в можливості доступу до нього з будь-якого місця, де є інтернет. Відмінність від нативного (локального) застосунку навряд чи буде помітна. Для використання "Easy symbol" рекомендовано запустити його безпосередньо за допомогою вищевказаного сайта.

Бібліотека MSS підтримує різні стандарти символів, які отримують за допомогою генератора через сервісну архітектуру. Під час першого запуску вебзастосунку можна вибирати між “міжнародним” і “розширеним” стандартами. Вибір роблять, натискаючи на один з двох прапорців (рис. 2). “Міжнародний” стандарт відповідає визначенню у MIL-STD-2525C.

“Розширений” для застосунку до міжнародної бази стандарт також охоплює підписи і тактичну графіку, яку використовують лише швейцарські збройні сили. Стандарт символу можна змінити в будь-який час у вікні параметрів “Easy Symbol”.

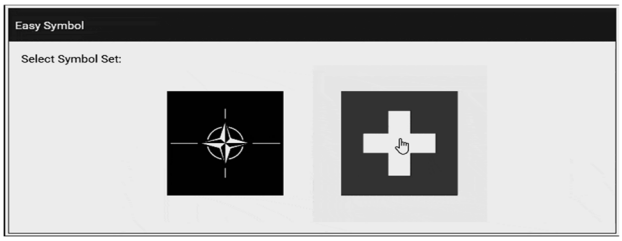


Рис. 2. Вибір стандартів

Діалог пошуку відкривають натисканням на іконку пошуку в головному вікні. У діалоговому вікні можна виконати пошук потрібної тактичної графіки.

Додаткові фільтри призначені для спрощення пошуку потрібного символу. Це передбачає фільтрацію за групами символів і активацію/деактивацію “Чутливості до регістра” і кнопку “Увімкнути символи N-точок”. Пошук символу починають, натискаючи “Return” або кнопку “Search”. Пошук розширених термінів також підтримується. Наприклад, можна шукати тип обладнання (“F/A 18”). Результати пошуку відображаються у правій частині вікна (рис. 3). У списку результатів символи відображаються разом із ім’ям символу і скороченням. Результати можна додатково відфільтрувати. Фільтр (символ воронки) доступний над областю результатів.

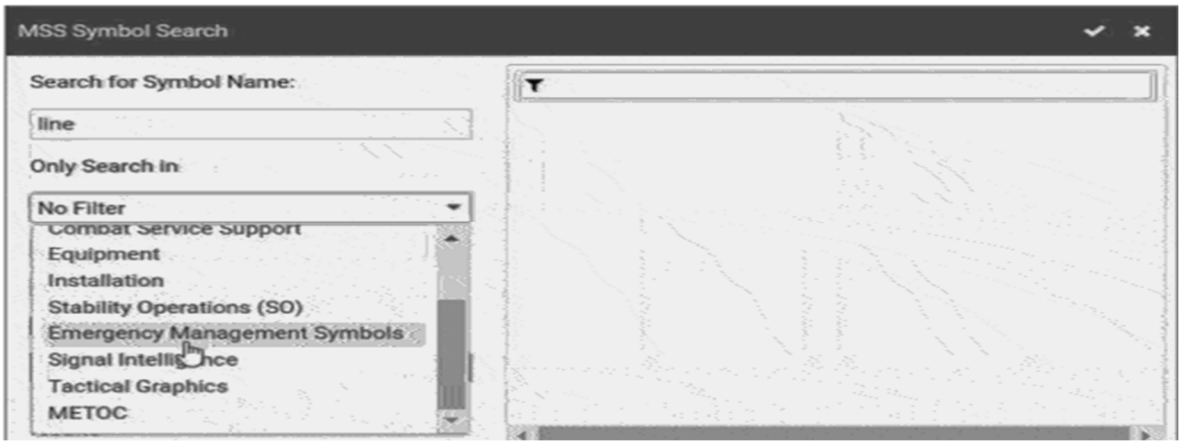


Рис. 3. Пошук символу

Залежно від варіанта використання військові символи відображаються по-різному. Прикладом може слугувати поліпшена видимість символів на карті. Символи MSS відповідають міжнародним правилам, визначеним у стандартах символіки. Але, крім того, MSS була оптимізована на основі відгуків партнерів і користувачів. Всі ці коригування не обов’язкові, їх можна вимкнути в будь-який час, щоб підписи MSS відповідали національним і міжнародним стандартам [2]. Ці налаштування контролюються параметрами формату. Також програма містить додаткові можливості для динамічного створення документації з використанням бібліотеки MSS. Для цього потрібно відкрити спеціальний застосунок MssDocumentor. Його розробила та сама команда, що успішно створила MIR

GEN, який сьогодні використовує НАТО (Програма військової сумісності) для створення визначень відображень символіки. Робота застосунку починається з бази даних або документа Excel. Цей базовий документ містить структуру стандарту і передбачає визначення глав у стандарті, а також ідентифікатори символів (рис. 4).

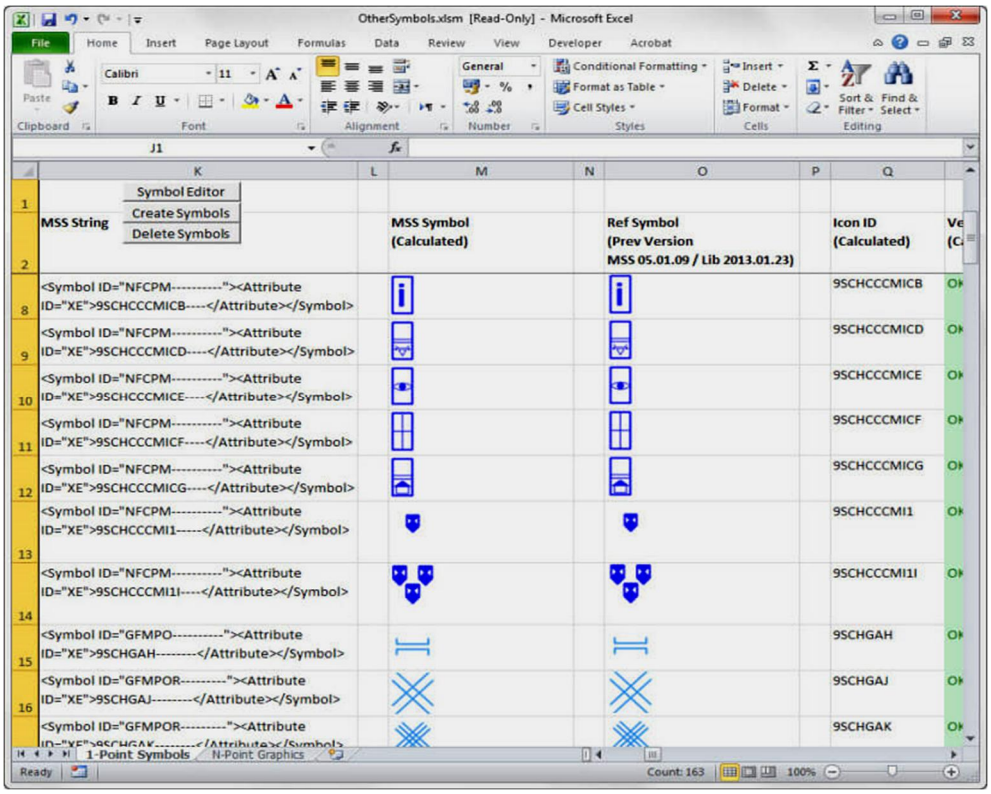


Рис. 4. Приклад базового документа

На першому етапі базовий документ завантажується у MssDocumentor. Під час імпорту дані перевіряють. Параметри рендерингу, такі як “ширина лінії”, “вибір кольору”, “представлення символу”, задають на другому етапі. Крім того, можна встановити такі параметри, як формат виведення мови (англійська, французька, німецька, італійська). На третьому кроці потрібно визначити, як відображатиметься тактична графіка. Можна вибирати між коридорним або некоридорним поданням. Крім того, можна відображати або приховувати контрольні точки, які використовують для створення тактичної графіки. Генерацію документа можна розпочинати одним кліком миші після задавання визначень на першому – третьому кроці.

Переваги генератора умовних знаків “Easy symbol”:

· Можливість доступу до вебзастосунку з будь-якого місця, де є доступ до інтернету.

· Зручний пошук та великий об’єм модифікаторів і атрибутів, відповідно до визначених стандартів.

· Кожен модифікатор і кожен атрибут мають текст підказки.

· Можливість збереження та передавання значка через посилання.

Недоліки генератора:

· Немає підтримки української або російської мов.

· Не існує десктопної версії.

· Безкоштовна версія істотно обмежена в можливостях.

ArcGIS побудований навколо бази геоданих, яка використовує підхід об’єктно-реляційної бази даних для зберігання просторових даних. База геоданих – це “контейнер” для зберігання наборів

даних, що пов'язують просторові об'єкти з атрибутами [8]. База геоданих також може містити інформацію про топології та моделювати поведінку об'єктів, таких як перетин доріг, з правилами щодо того, як об'єкти пов'язані один з одним. Працюючи з базами геоданих, важливо розуміти класи об'єктів, що являють собою набір об'єктів, поданих точками, лініями або полігонами. Із шейп-файлами кожен файл може опрацьовувати тільки один тип об'єкта. База геоданих може зберігати кілька класів об'єктів або типів об'єктів у одному файлі. Бази геоданих в ArcGIS можуть зберігатися трьома різними способами – у вигляді "файлової бази геоданих", "персональної бази геоданих" або "бази геоданих ArcSDE" [5]. Файлова база геоданих зберігає інформацію у папці із розширенням .gdb. Нутрощі виглядають аналогічно покриттю, але насправді не є покриттям. Як і в персональній базі даних, файлова база геоданих підтримує тільки один редактор. Однак, на відміну від персональної бази геоданих, обмеження за розміром практично немає. За замовчуванням кожна окрема таблиця не повинна перевищувати 1 ТБ, але це можна змінити. Персональні бази геоданих зберігають дані в файлах Microsoft Access, використовуючи поле BLOB для зберігання даних геометрії. OGR Бібліотека може опрацьовувати файли цього типу, конвертувати їх в інші формати файлів.

Завдання адміністрування бази даних для персональних баз геоданих, такі як управління користувачами і створення резервних копій, можуть бути виконані через ArcCatalog. Персональні бази геоданих, основані на Microsoft Access, працюють тільки в Microsoft Windows і мають обмеження в 2 гігабайти. Бази геоданих рівня підприємства (на багато користувачів) опрацьовуються за допомогою ArcSDE, який взаємодіє із високопродуктивними СУБД, такими як PostgreSQL, Oracle, Microsoft SQL Server, DB2 і Informix для управління аспектами управління базами даних, тоді як ArcGIS займається управлінням просторовими даними. Бази геоданих рівня підприємства підтримують реплікацію бази даних, управління версіями і управління транзакціями, а також є кросплатформними і можуть працювати в Linux, Windows і Solaris.

Створена також персональна база даних SDE, яка працює з SQL Server Express. Особисті бази даних SDE не підтримують багатокористувацьке редагування, але підтримують управління версіями і вимкнене редагування. Microsoft обмежує бази даних SQL Server Express обсягом 4 ГБ.

Програмне забезпечення ArcGIS для побудови тактичної символіки використовує бібліотеку вбудованих компонентів геоінформаційних систем ArcGIS Engine. Вона призначена для розробників, які створюють для користувача застосунки. Використовуючи ArcGIS Engine, розробники можуть вбудовувати функції ArcGIS в інші інформаційні інструменти та створювати власні програми, які надають розширені ГІС-рішення. ArcGIS Engine підтримується у Windows, Solaris і Linux (Intel); тому розробники можуть створювати кросплатформні індивідуальні рішення для широкого кола користувачів. Основні функції ArcGIS Engine можна розділити на категорії:

· Базові сервіси – основні об'єкти ГІС ArcObject, які потрібні практично для будь-яких ГІС-застосунків, таких як геометрія об'єктів і відображення.

· Доступ до даних – ArcGIS Engine забезпечує доступ до широкого спектра растрових і векторних форматів, ураховуючи потужність і гнучкість бази геоданих.

· Подання карти – ArcObject для створення і відображення карти з можливостями символіки, написів і тематичних карт, ураховуючи користувацькі застосунки.

· Компоненти розробника – управління призначеним для користувача інтерфейсом високого рівня для швидкого розроблення застосунків, а також комплексна довідкова система і приклади інструментів для ефективного розроблення.

· Розширення – ArcGIS Engine Runtime можна розгорнути зі стандартною функціональністю або з додатковими розширеннями для більшої функціональності.

Два основні компоненти ArcGIS Engine, ArcGIS Engine Software Developer Kit (SDK) забезпечують платформу для усіх застосунків ArcGIS. SDK – це набір інструментів для розробників застосунків. Він надає візуальні компоненти, приклади, інструменти, майстри, шаблони, інтерфейси

прикладного програмування (API) і розділи довідки, які допомагають розробникам створювати складні ГІС-застосунки. Всі застосунки, створені з використанням ArcGIS Engine SDK, вимагають, щоб ArcGIS Engine Runtime або ArcGIS Desktop були встановлені з відповідною ліцензією. ArcGIS Engine Runtime – це платформа, на якій побудований ArcGIS Desktop; це дає змогу користувачам застосунків ArcGIS Desktop запускати призначені для користувача програми на основі ArcGIS Engine. ArcGIS Engine забезпечує базову функціональність усіх застосунків ArcGIS, таких як взаємодія із картами, створення карт, аналіз карт, створення даних (файл форми й особиста база геоданих), елементи керування для розробників, а також технології для розробників і геоопрацювання. Крім того, додаткові можливості прикладного програмування, такі як оновлення бази геоданих, просторова, тривимірна, мережева схеми відстеження або сумісність даних, надають як конкретні розширення.

Розширення бази геоданих. Розширення бази геоданих для ArcGIS Engine Runtime додає можливість створювати й оновлювати дані користувачів, за допомогою ArcSDE, або файлову базу геоданих. Передбачає можливість роботи зі схемами і версіями бази геоданих. Розширення бази геоданих розблокує ArcGIS Engine Runtime з необхідними об'єктами ArcObject для запуску користувальницького редагування і розширених рішень бази геоданих. Ці рішення містять застосунки, які займаються автоматизацією і компіляцією даних ГІС, а також створенням і обслуговуванням функцій бази геоданих. Розширення бази геоданих дає можливість програмно визначати поведінку бази, а саме версії, лінійні мережі, топології, підтипи і геометричні мережі.

Просторове розширення. ArcGIS Engine Runtime Spatial надає потужний набір функцій, які дають змогу застосункам створювати, зчитувати й аналізувати растрові дані на основі осередків. Цей тип аналізу дає можливість користувачам отримувати інформацію про свої дані, визначати просторові відносини, знаходити відповідні місця розташування і розраховувати сукупну вартість переміщення з однієї точки в іншу. Інші розширені застосунки, які підтримує це розширення, передбачають розрахунок нахилу і контурів за цифровими моделями рельєфу (ЦМР). 3D-розширення для ArcGIS Engine Runtime дає змогу також візуалізувати дані в трьох вимірах. Це розширення доповнює стандартний ArcGIS Engine компонентами для перегляду поверхні з декількох точок огляду і визначення того, що видно з вибраного місця розташування. SceneControl і GlobeControl надають інтерфейс для перегляду декількох 3D-шарів і глобальних даних для візуалізації, створення і аналізу поверхонь.

Розширення мережі доповнює стандартне середовище виконання ArcGIS Engine, додаючи можливість маршрутизації, аналізу області обслуговування, а також створення наборів мережевих даних і управління ними. Розширення мережі дає змогу розробникам створювати і розгортати потужні користувальницькі застосунки для транспорту, реагування на надзвичайні ситуації, протипожежні, військові та для інших цілей. Розширення Maplex дає змогу розробникам використовувати механізм розміщення міток Maplex для високоякісного розташування тексту і самих міток: картками, створеними за допомогою ArcGIS Desktop, можна виділяти на мапі потрібні області. Розширення відстеження підтримує відображення, аналіз і маніпулювання тимчасовими даними в рішенні ArcGIS Engine. Розширення надає компоненти розробника і ArcObjects для управління схематичними даними і процесами. Розширення Schematics для ArcGIS Engine підтримує аналіз, відображення і маніпулювання даними схеми. Розширення Data Interoperability усуває бар'єри для спільного використання даних, надаючи прямий доступ до даних, їх перетворення і експорт за допомогою інструментів геоопрацювання. Це розширення дає змогу застосункам ArcGIS Engine використовувати і поширювати дані в багатьох форматах. Також існує служба військових символів (MSS), яка теж є базою даних умовних знаків [3]. MSS (Military Symbol Service) – дуже потужна і компактна бібліотека військових символів і тактичної графіки.

Сама бібліотека сьогодні містить понад 3500 символів і тактичної графіки, які визначені в певних національних і міжнародних правилах (табл. 1).

Таблиця 1

Міжнародні правила нанесення тактичної графіки

Тип умовного стандарту	Назва регламенту
НАТО	APP-6 (B) та MIL-STD-2525C
Швейцарські збройні сили	REGL.52.002.03 / 04
Поліція	Führung im Polizeieinsatz
Правила дорожнього руху	Regl V + TCS: Strassensignale

MSS реалізують у деяких системах такі постачальники: TADIRAN, EADS, THALES, AMPER, ELTA, ELCA. Його використовують на національному рівні Швейцарські збройні сили і на міжнародному рівні НАТО (JC3IEDM/MIP) як постачальника символів і тактичної графіки. Символи або тактична графіка не замінюються зображеннями; їх опрацьовують через SymbolString. Це дає змогу швидко обмінюватись графікою лише за кілька символів. Бібліотека MSS, яка містить всі символи і тактичну графіку, може бути доповнена графікою для конкретної країни або продукту. Це передбачає також параметри відображення і налаштування. Сервіс може бути реалізований в будь-якій системі, що потребує військової символіки та тактичної графіки, та надає широкий спектр форматів експорту.

Стандарт умовних знаків застосовують для електронних (автоматизованих) систем та нанесення графічних знаків від руки, як кольорових, так і монохромних. Їх потрібно використовувати під час нанесення обстановки на картах, схемах. У Стандарті умовних знаків наведено модульні блоки для стандартизації складання умовних знаків. Блоками є обрамлення, умовні позначення, ампліфікатори та модифікатори, які наносять з використанням кольору, графіки та алфавітно-цифровим способом. У Стандарті умовних знаків встановлено деталізовані стандарти та обов'язкові вимоги для побудови умовних знаків із відповідним ступенем гнучкості для потреб окремих користувачів. Набір умовних знаків призначено для графічного відображення підрозділів, озброєння, інфраструктури та інших елементів, а також діяльності, що стосується міжвидових операцій (бойових дій) [6]. Ці знаки містять складові елементи для створення міжвидових умовних знаків за сферами застосування: сухопутної, повітряної, морської, космічної, відображення стабілізаційної діяльності для підтримання цивільного населення.

Кожен графічний набір знаків, спрямований на підтримання цивільного населення та управління військами (силами), наведено нижче. На рис. 5 подано структуру міжвидових умовних знаків, що використовують під час планування та ведення міжвидових операцій.

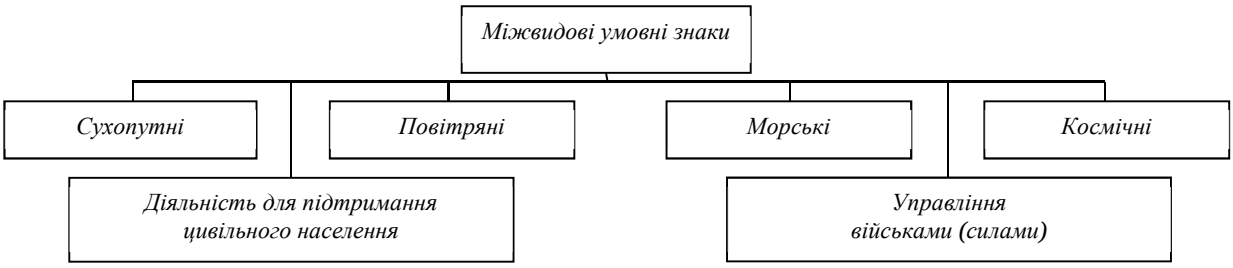


Рис. 5. Структура міжвидових умовних знаків

Основне призначення міжвидових умовних знаків – графічне передавання інформації про об'єкт, що відображається.

Під військовими об'єктами потрібно розуміти фізичні об'єкти (бойові позиції військ (підрозділів), озброєння та військову техніку, пункти управління, полігони, вузли зв'язку, радіотехнічні системи, бази, склади, об'єкти життєзабезпечення (інфраструктуру) військ (підрозділів) та нефізичні об'єкти (планування, заходи контролю, очікувані місця з тимчасово визначеними характеристиками).

Крім того, знаки також використовують для відображення діяльності, спрямованої на підтримання цивільного населення. У Стандарті умовних знаків зосереджено знаки для використання у сучасних мультихроматичних електронних системах [5, 6]. Водночас усі знаки можна застосовувати у монохроматичних системах та для нанесення обстановки від руки. Потреба у зменшенні засмічення

екранів зайвою інформацією визначає вимоги до варіантів знаків, що передбачають можливість зменшення їх розмірів та обсягу іншої інформації про знаки.

Деякі позначки підрозділів складні та можуть займати весь умовний знак. Іноді їх складові накладаються одна на одну. Тому у деяких випадках необхідно зменшити їх розміри, забезпечивши можливість їх візуального розпізнавання. На рис. 6 наведено співвідношення секцій умовного восьмикутника для забезпечення максимальної візуалізації.

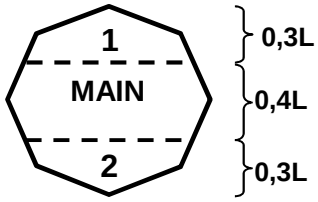
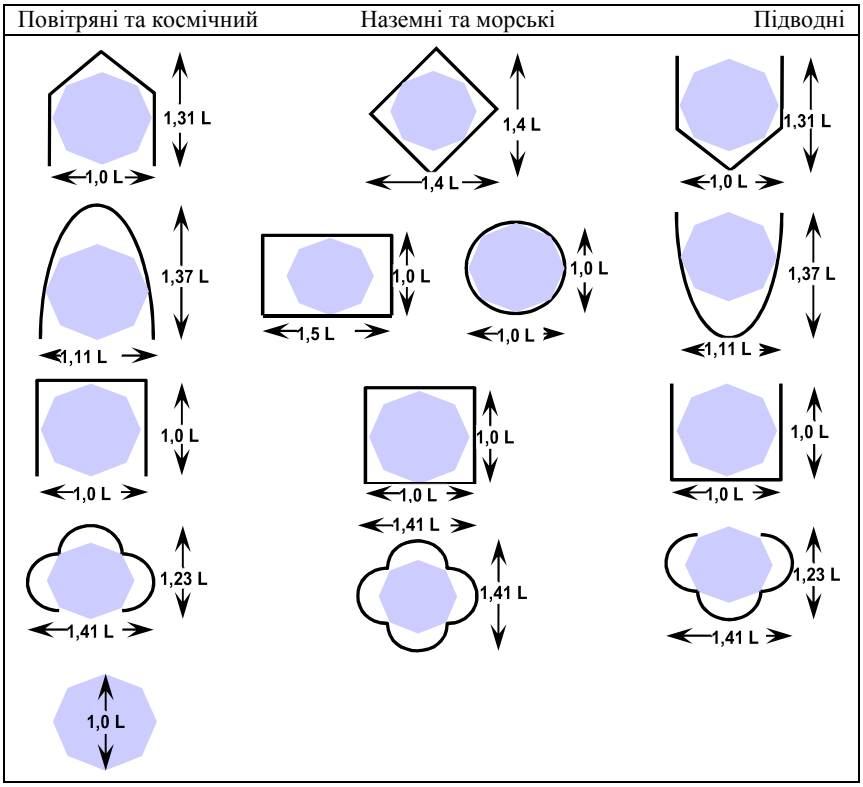


Рис. 6. Співвідношення розмірів секцій умовного восьмикутника

Відповідні розміри кожного умовного знака та його компоненти повинні встановлюватися відповідно до визначеної специфікації. Кожен із цих розмірів необхідно витримувати згідно із пропорціями. Розмір умовного знака має визначатися щодо умовного восьмикутника, розміщеного всередині його обрамлення. Відносно величини “L” визначають довжину та висоту восьмикутника. Висота та довжина можуть змінюватися від 1,0L до 1,5L залежно від форми обрамлення. Мінімальний розмір точки – 0,15L. Загалом позначка не має бути більшою за внутрішнє обрамлення знака. Винятком є позначки, що займають увесь внутрішній сектор умовного знака та в будь-якому випадку повинні торкатися його внутрішнього обрамлення. Розміри умовних знаків з неповним обрамленням (повітряні та підводні) мають відповідати розмірам знаків з повним обрамленням (наземні, надводні).

Таблиця 2

Співвідношення розмірів умовних знаків



Метод формування умовних знаків, що побудовані за допомогою символів, запроваджує логічно структурований підхід до створення знака. Окремі графічні функції або атрибути відібрано для зображення кожного окремого типу об'єкта в оперативному середовищі з відповідними ознаками, притаманними конкретному знаку. Наприклад, позначкою для вертолітного підрозділу є графічні "пропелери". Підхід, запроваджений у цьому стандарті, на відміну від жорстко визначеної стилізованої піктограми знака, дає змогу в разі потреби тимчасово змінювати позначку для максимальної її візуалізації на екрані монітора. Оскільки обрамлення знака визначає стандартну тотожність і розмір об'єкта, дуже важливо, щоб ширина лінії була достатньою для забезпечення чіткості зображення за нормальної відстані до знака, який розглядають. Оптимальна ширина лінії обрамлення може відрізнятися залежно від типу, розміру та бути заповненою або незаповненою, відображатися в кольорі або чорно-білою лінією. Зважаючи на усе це, необхідно перевіряти оптимальну візуалізацію та вибирати товщину лінії обрамлення знака від 0,75 до 2,25 мм.

Під час проєктування та розроблення знаків, складання композиції їх складових необхідно ураховувати психологічні чинники, такі як здатність легко розпізнати знаки, їх розбірливість у різних умовах освітлення на різноманітній картографічній основі, на електронних екранах різних розмірів і типів за різних ступенів фізичної та інтелектуальної втоми.

Формулювання мети та постановка задачі

Підготовчий етап визначення цілей передбачає усвідомлення важливості їх формування, оскільки однією з найпоширеніших помилок, які істотно знижують ефективність вибору стратегії, є трактування цього процесу як очевидного, формального. Багато організацій не мають чітко визначених і задокументованих цілей. Дослідження показало, що успішні компанії витрачають 90 % часу на мотивування своїх працівників до досягнення цілей і 10 % на формулювання цілей. Водночас, за відсутності конкретних цілей, 90 % часу витрачають на створення і затвердження різних правил, процедур і методів. Постановка цілей – доволі трудомісткий і відповідальний процес, що складається із таких основних етапів:

- виявлення й аналіз трендів у середовищі;
- вибір цілей проєкту загалом;
- побудова ієрархії цілей;
- визначення індивідуальних цілей.

Гнучкість цілей передбачає їх уточнення відповідно до змін, які відбуваються в навколишньому середовищі, хоча це не означає, що вони повністю залежать від нього. Виявлення змін у середовищі, характерних для процесів економічного розвитку, соціально-політичної сфери, науки і техніки, дає змогу визначити цілі відповідно до передбачення. Виявлені тенденції не повинні бути абсолютизовані, але їх потрібно урахувати.

Цікавий підхід запропонував Ф. Котлер, який радить відібрати цілі, починаючи з фінансової, і на основі формулювання маркетингу [12]. Варто узгодити цілі так: визначити бажаний рівень чистого прибутку; розрахувати суму доходу, необхідну для досягнення цього результату; на основі середньої ціни продажу визначити відповідний фізичний обсяг продажів; з урахуванням очікуваного рівня світового попиту розрахувати необхідну частку ринку, сформулювати цілі у сфері продажів і комунікацій. Однак такий підхід передбачає високий рівень інформаційного забезпечення. В іншому випадку необхідно сформулювати різні гіпотези та альтернативні плани подальшої перевірки чутливості результатів розрахунків до сформульованих гіпотез.

Існує певна сукупність цілей, характерних для складної організації, тому вони повинні бути запрограмовані, тобто потрібно досягти головної мети, сформувати цілі другого рівня, досягти їх, а також цілей третього рівня, проміжних та підцілей.

Основні цілі, визначені для довгострокової перспективи, відповідають концепції розвитку організації та її основним напрямам. Водночас ранжування цілей ґрунтується на принципі пріоритету: забезпечення максимальної рентабельності зі збереженням діяльності; забезпечення стійкості позиції організації; розроблення нових напрямів розвитку (діяльності).

Із урахуванням основних (корпоративних) цілей послідовність розроблення конкретних цілей така: цілі конкретних СГЦ (внутрішньокорпоративні організаційні одиниці); цілі розвитку функціональних сфер (маркетинг, виробництво, фінанси, персонал, наукові дослідження тощо); цілі філій та дочірніх компаній. Корпоративні цілі стосуються корпорації загалом як наслідку і реального втілення його місії. Вони повинні дотримуватися цілей окремих бізнес-одиниць.

Функціональні цілі багато в чому випливають з цілей корпорації та її СГЦ. Функціональні одиниці мають свої цілі, пов'язані з їхньою безпосередньою діяльністю. Створення ієрархії передбачає формування таких цілей, досягнення яких на певних рівнях визначає реалізацію корпоративних цілей [5]. Отже, досягнення головної мети починається із останньої підцілі, і кожен перехід до наступної потребує виконання попередньої. Формування "дерева" передбачає уточнення цілей відповідно до конкретного ринку та ролі, яку він прагне виконувати на ринку. Необхідно також враховувати, що ця позиція буде досягнута (економія витрат, диференціація або концентрація). На цьому етапі йдеться лише про загальну орієнтацію, яка буде втілена в конкретну стратегію для кожного СГЦ.

Під час декомпозиції цілей повинна забезпечуватись узгодженість між довго-, середньо- та короткостроковими цілями, різними видами діяльності (виробництво та маркетинг, виробництво та фінанси тощо). Для того, щоб ієрархія цілей в організації стала логічно повною і реальним ефективним інструментом їх досягнення, вона повинна бути доведена до конкретного виконавця.

Отже, визначення цілей є трудомістким, але об'єктивно необхідним етапом стратегічного планування, оскільки цілі – критерій прийняття рішень про необхідність змін, згортання або розширення ринкової політики організації. Схема (рис. 7) характеризує поділ загальної мети створення програмного забезпечення на складові елементи.

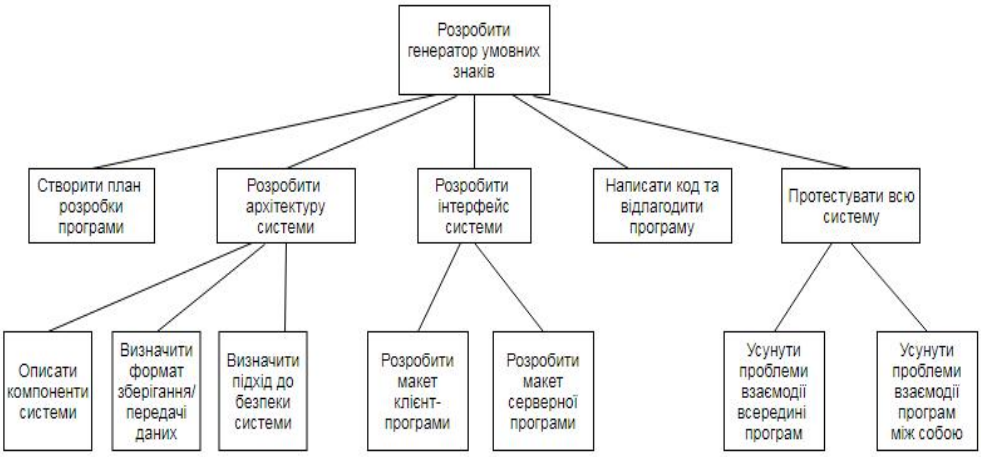


Рис. 7. Дерево цілей генератора умовних знаків

Виклад основного матеріалу

UML – не просто графічна мова. Кожній частині системи графічної нотації відповідає *специфікація*, що містить текстове представлення синтаксису і семантики відповідного будівельного блока. Наприклад, піктограмі класу відповідає специфікація, яка повністю описує її атрибути, операції (ураховуючи повні сигнатури) і поведінку, хоча візуально піктограма іноді відображає тільки малу частину цієї сукупності. І навіть більше, може існувати інше уявлення цього класу, що відображає зовсім інші його аспекти, проте є відповідним все до тієї ж специфікації [25]. Специфікація класу може містити й інші деталі, наприклад видимість атрибутів і операцій або вказівку на те, що клас є абстрактним [13]. Багато таких деталей можна візуалізувати у вигляді графічних або текстових доповнень до стандартного прямокутника, зображенням класу.

Спочатку потрібно визначити мету моделювання. Мета моделювання – опис розроблення генератора умовних знаків. Основним управлінським механізмом є правила стандартизації умовних

знаків, згідно зі стандартами APP-6D. Як вхідні дані використано необхідну літературу для створення програми. Виконавці (механізми): Програмне забезпечення; База даних символів. На виході цього процесу повинно бути готове програмне забезпечення для генерування умовних знаків.

На рис. 8 зображено кілька груп користувачів (actors), які працюватимуть з інтелектуальною інформаційною системою. Спочатку вибирають загальний тип всіх користувачів (узагальнення), відтак усіх поділяють на два типи: користувач сервера і користувач генератора. Другий тип поділено ще на шість класів, із загальною характеристикою (створення умовного знака), але різними привілеями і функціями (рис. 8). Для візуалізації, конструювання та документування програмних систем необхідно розглядати їх з різних позицій. У всіх причетних до проєкту, – а це кінцеві користувачі, аналітики, розробники, системні інтегратори, тестувальники, технічні кодери і менеджери проєктів – є власні інтереси, і кожен розглядає створювану систему по-різному в різні моменти її життя [13]. Системна архітектура є, мабуть, найважливішим артефактом, який використовується для управління всілякими поглядами і тим самим сприяє розробленню системи впродовж всього її життєвого циклу.

Архітектура – це сукупність істотних рішень щодо:

- організації програмної системи;
- вибору структурних елементів, що утворюють систему, та їх інтерфейсів;
- поведінки цих елементів, специфікованих у кооперациях з іншими елементами;
- складання із цих структурних і поведінкових елементів все більших підсистем;
- архітектурного стилю, що спрямовує і визначає всю організацію системи: статичні та динамічні елементи, їх інтерфейси, спосіб їх кооперації та об'єднання.

Архітектура програмної системи охоплює не тільки її структурні та поведінкові аспекти, а й використання, функціональність, продуктивність, гнучкість, можливості повторного застосування, повноту, економічні та технологічні обмеження і компроміси, а також естетичні питання.

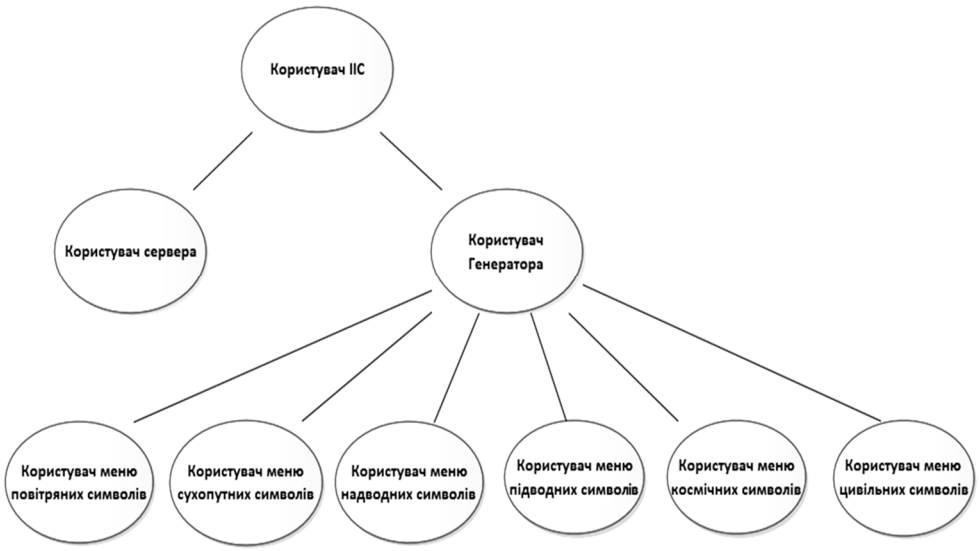


Рис. 8. Діаграма ієрархії користувачів

Вид з погляду прецедентів (Use case view) охоплює прецеденти, що описують поведінку системи, яку спостерігають кінцеві користувачі, аналітики і тестувальники. Цей вид специфікує не істину організації програмної системи, а ті рушійні сили, від яких залежить формування системної архітектури. У мові UML статичні аспекти цього виду передають діаграмами прецедентів, а динамічні – діаграмами взаємодії, станів і дій. Приклад використання описує типову взаємодію між користувачем та системою. У найпростішому випадку варіант використання визначають, обговорюючи із користувачем функції, які він хотів би реалізувати. Діаграму варіантів використання подають у вигляді графа, вершини якого – UML кола, а стрілки – відносини [21]. Розглянемо діаграми використання основних класів користувачів для кращої деталізації під час розроблення (рис. 9–13).

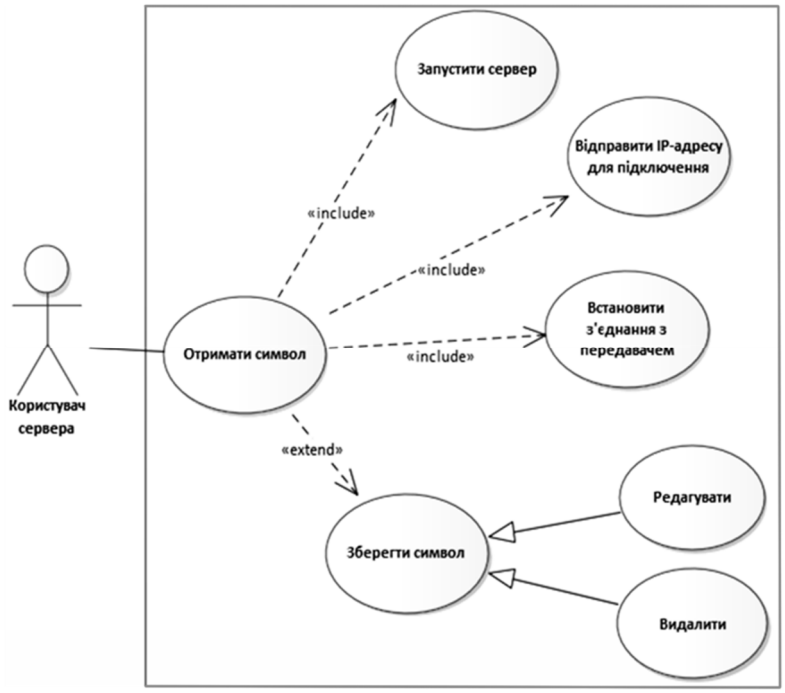


Рис. 9. Діаграма варіантів для класу користувачів "Користувач сервера"

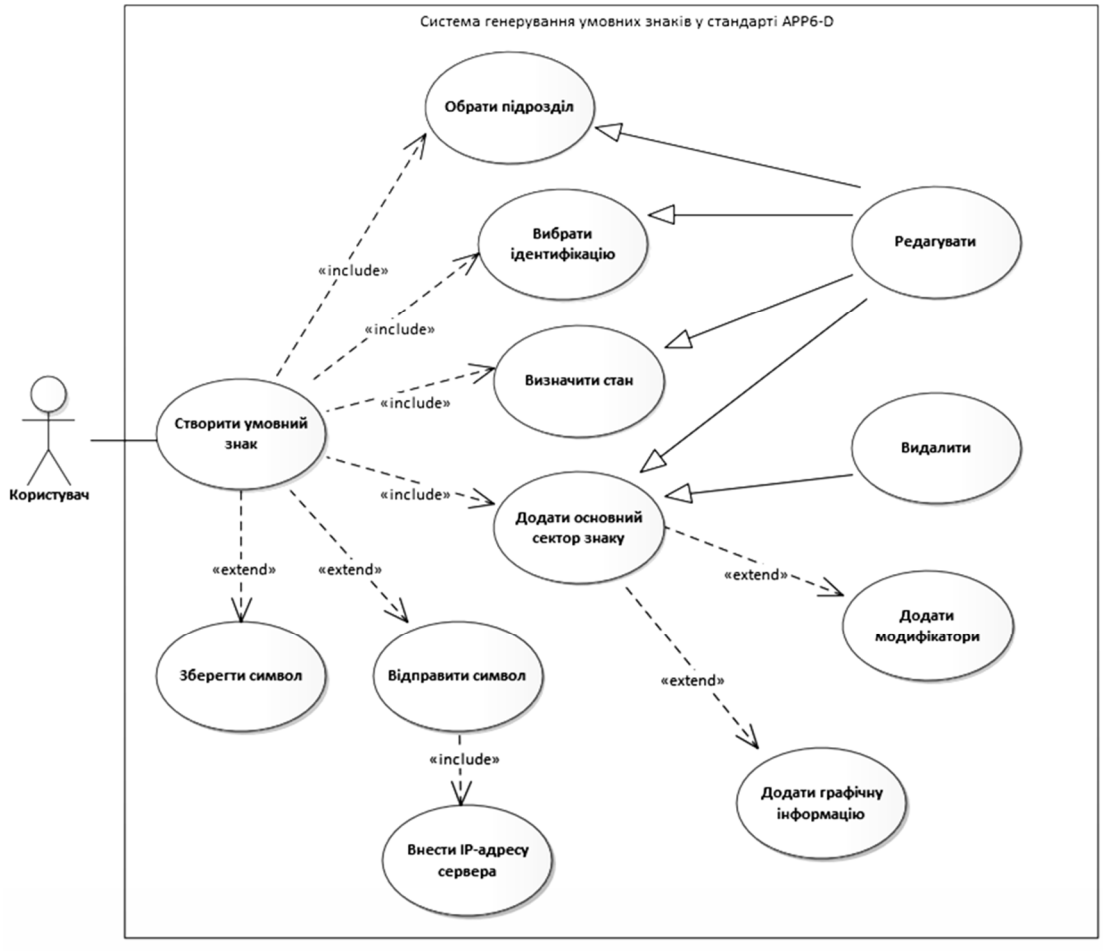


Рис. 10. Діаграма варіантів для класу користувачів "Користувач генератора"

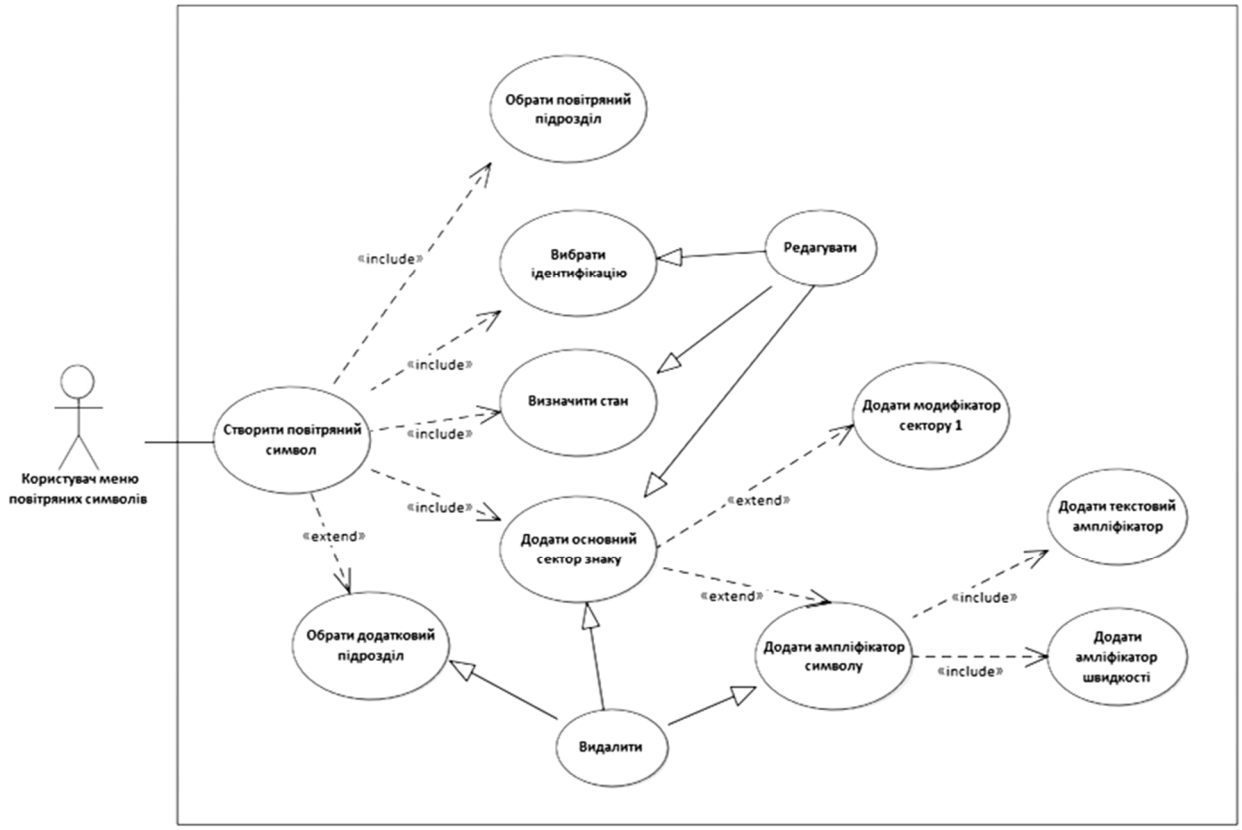


Рис. 11. Діаграма варіантів для класу "Користувач меню повітряних символів"

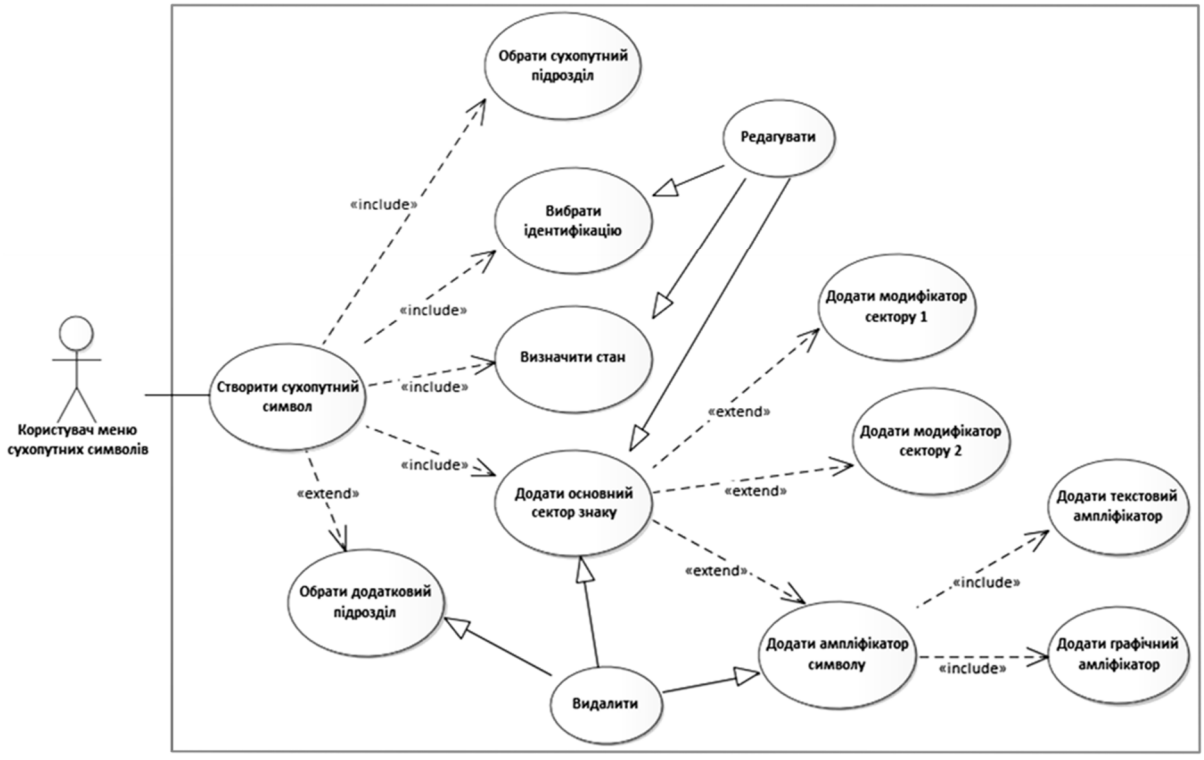


Рис. 12. Діаграма варіантів для класу "Користувач меню сухопутних символів"

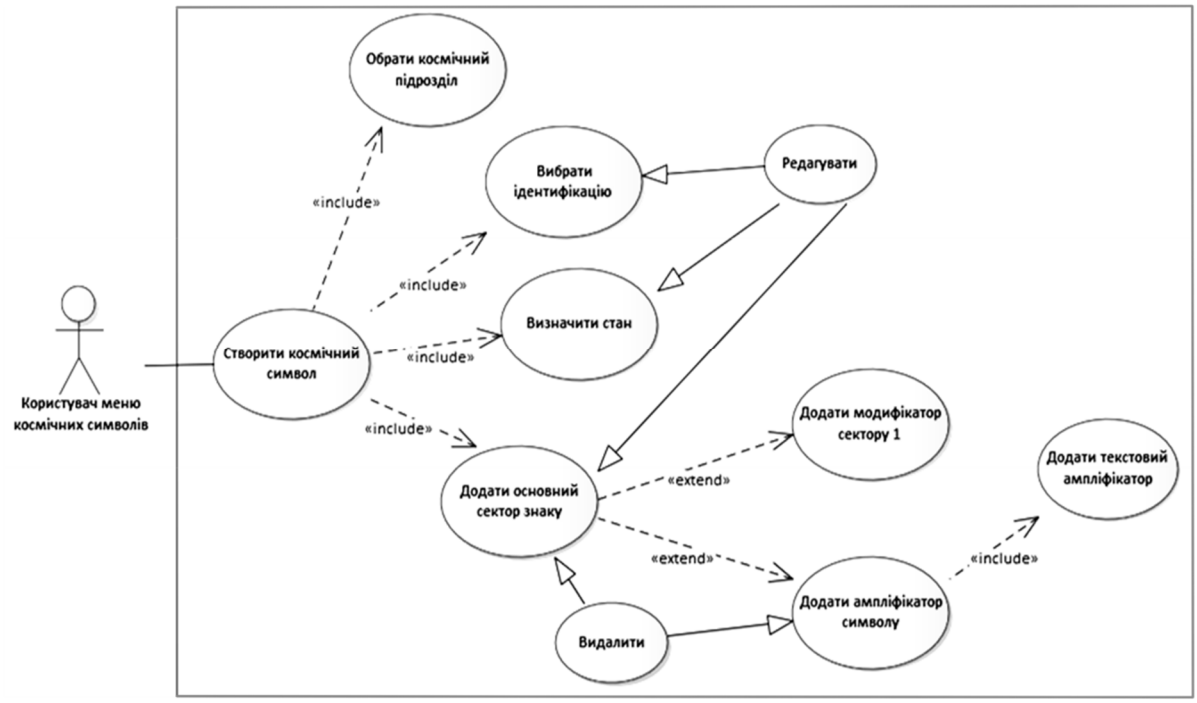


Рис. 13. Діаграма варіантів для класу "Користувач меню космічних символів"

Вигляд з погляду проєктування (Design view) охоплює класи, інтерфейси і кооперації, що формують словник завдання і його рішення. Цей вид підтримує насамперед функціональні вимоги, що ставлять до системи, тобто ті послуги, які вона повинна надавати кінцевим користувачам [13]. За допомогою мови UML статичні аспекти цього виду можна передавати діаграмами класів і об'єктів (рис. 14).

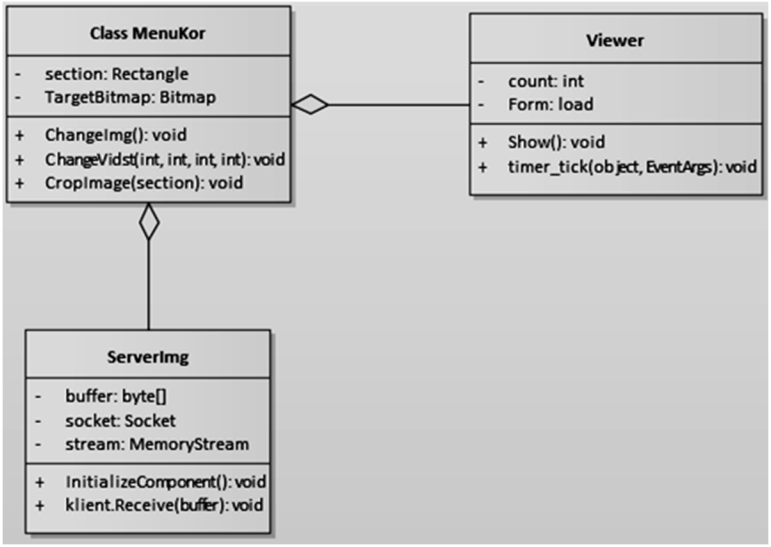


Рис. 14. Діаграма класів

Вигляд з погляду процесів (Process view) охоплює процеси, що формують механізми паралелізму і синхронізації в системі. Цей вигляд описує передусім продуктивність, масштабованість і пропускну здатність системи. В UML його статичні та динамічні аспекти візуалізують тими самими діаграмами, що і для вигляду з погляду проєктування, але особливу увагу звертають на активні класи, які відображають відповідні процеси (рис. 15).

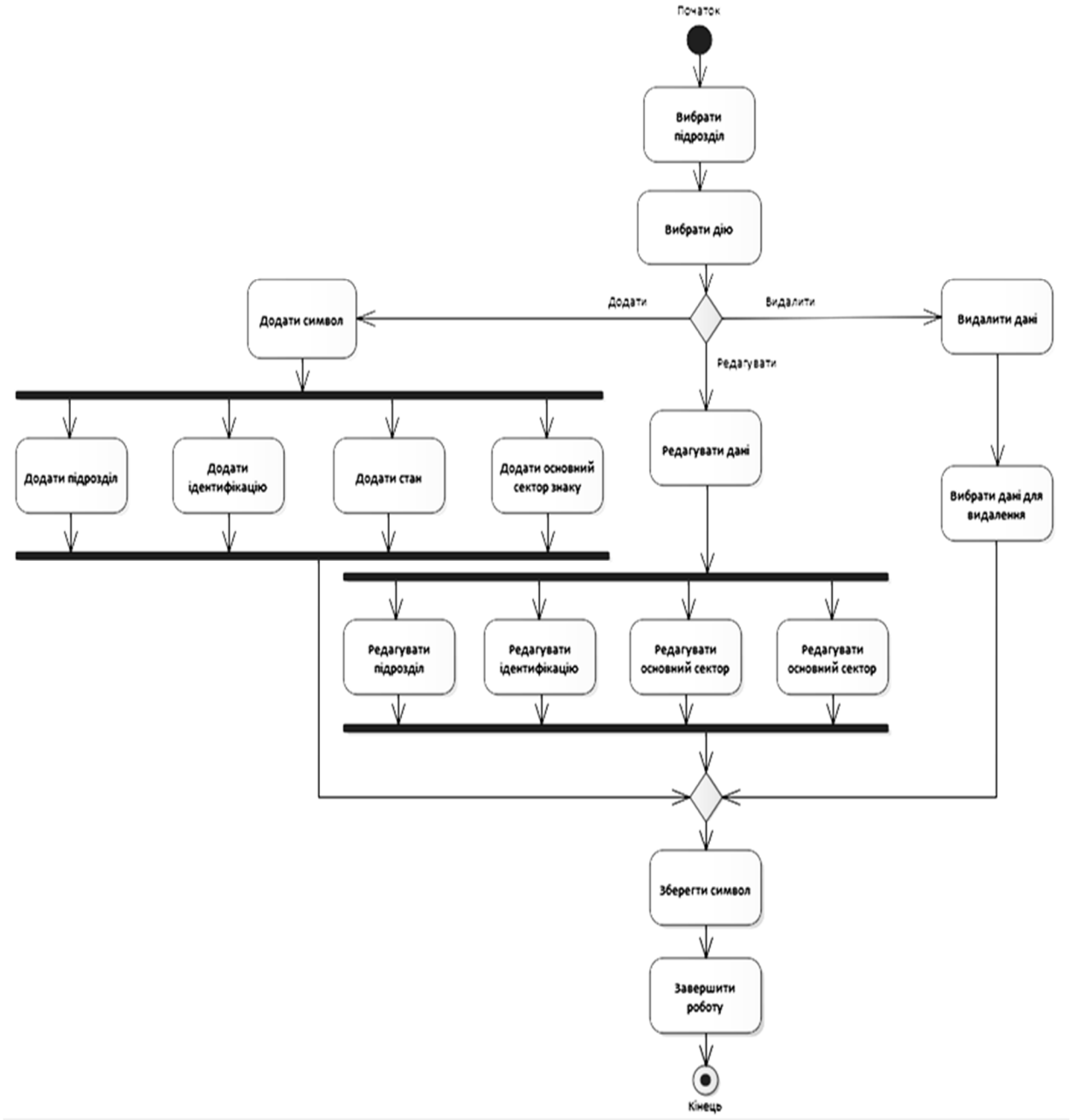


Рис. 15. Діаграма станів

Вигляд з погляду розгортання (Deployment view) охоплює вузли, що формують топологію апаратних засобів системи, на якій вона виконується. Він пов’язаний насамперед із розподілом, постачанням і встановленням частин, що становлять фізичну систему. Його статичні аспекти детально описують діаграмами розгортання (рис. 16).

Кожен із перелічених видів можна вважати цілком самостійним, так що особи, причетні до розроблення системи, можуть зосередитися на вивченні тільки тих аспектів архітектури, які безпосередньо їх стосуються [21]. Але не можна забувати про те, що ці види взаємодіють один з одним. Наприклад, вузли вигляду з погляду розгортання містять компоненти, описані для вигляду з погляду реалізації, а ті, своєю чергою, є фізичним втіленням класів, інтерфейсів, кооперацій та активних класів з виглядів під кутом зору проєктування і процесів. UML дає змогу відобразити кожний із п’яти перерахованих виглядів і їх взаємодії.

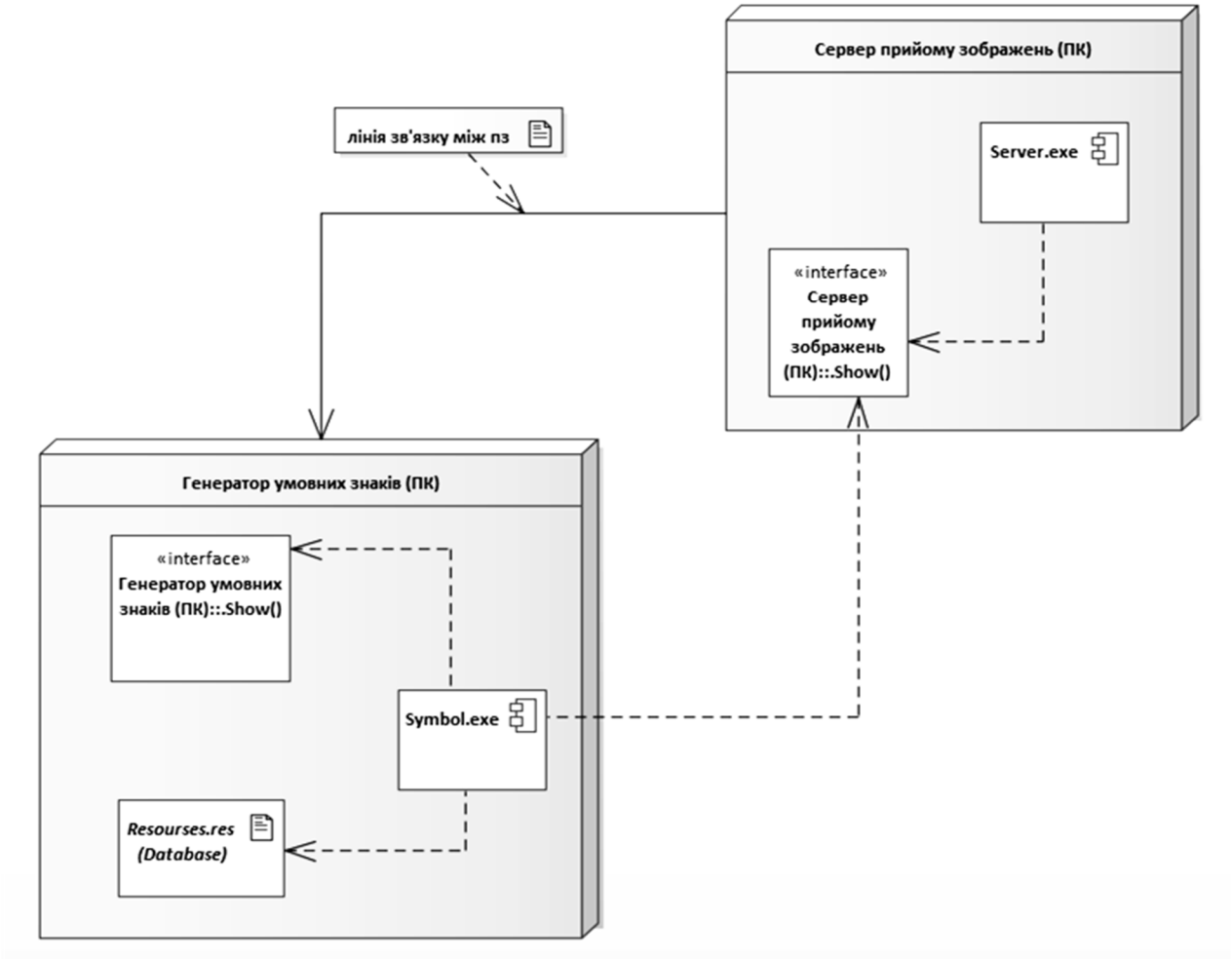


Рис. 16. Діаграма компонентів

Для конкретизації системи також можна використовувати ієрархії задач. Основне завдання побудови ієрархії задач полягає в тому, щоб зробити опис модельованої предметної області чіткішим і зрозумілішим на кожному рівні деталізації.

Ієрархія задач – це інтерфейс для перегляду дерева підпорядкованих задач. В ієрархії можуть бути зібрані задачі з різних категорій. Ієрархія може будуватись не лише за задачами, але і за будь-якими об'єктами системи – наприклад, за організаційною структурою, складом груп тощо. Однак основне її застосування – це робота із задачами. Ієрархія може містити не тільки звичайні, статичні стовпці, а й доповнюватися динамічними стовпцями.

Розроблення ієрархії розпочинають із кінцевих цілей та успішного розподілу її у керовані частини, які оцінені за критеріями розміру, тривалості та відповідальностей (наприклад, системи, підсистеми, компоненти, задачі, підзадачі та пакети робіт) та містять усі необхідні кроки для досягнення побудови системи.

Ієрархія задач надає загальний каркас для природного розвитку загального планування та контролю системою і є базисом для розподілу роботи у такі інкременти, які можуть бути визначеними та з яких можна створити технічне завдання й установлені звіти щодо технічних даних, графіків, вартостей, робочих годин [21].

Для кожного елемента структури декомпозиції робіт генерується опис задачі, яку потрібно виконати [25]. Ця техніка (іноді її називають структурою декомпозиції системи) використовується для визначення і налагодження сумарних рамок системи.

Структуру конкретної досліджуваної системи та чіткі ієрархічні рівні – групи елементів, розміщені на однаковій відстані (вимірюваній як кількість ребер) від головного елемента (кореня дерева), у вигляді ієрархії процесів задач різних рівнів подано на рис. 17.

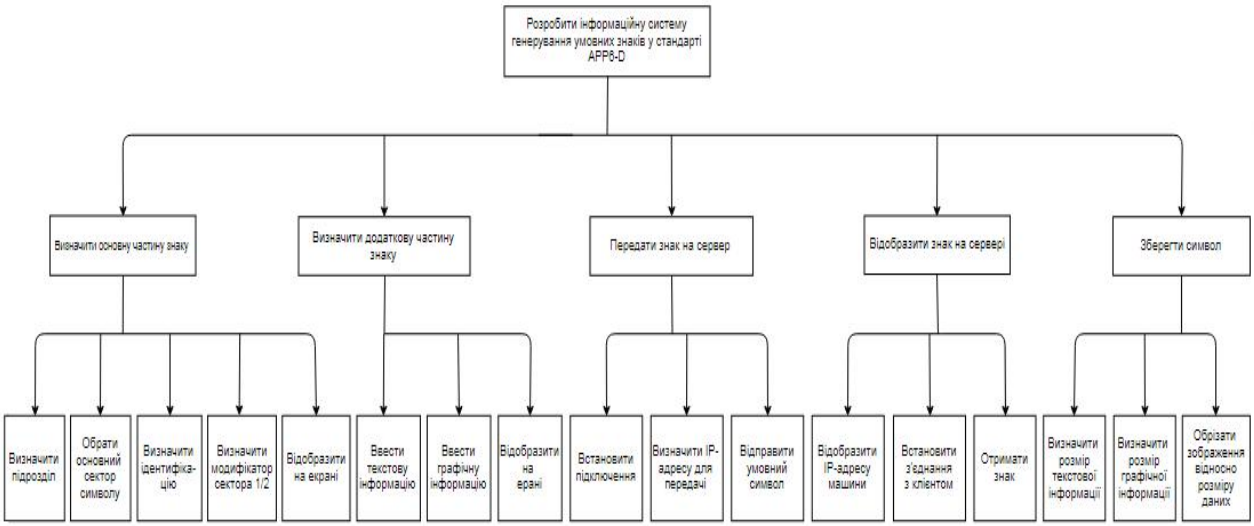


Рис. 17. Ієрархія задач

Для виконання поставленого завдання вибрано таке середовище розроблення, як Microsoft Visual Studio 2015 [1–3, 8–12]. Microsoft Visual Studio – це серія продуктів корпорації Майкрософт, ураховуючи інтегроване середовище розроблення програмного забезпечення та низку інших інструментів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, зокрема підтримку технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у власних, так і в керованих кодах для всіх платформ, підтримуваних Microsoft Windows, Мобільний, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework і Microsoft Silverlight.

В інформаційній системі використано такі основні елементи керування:

1. Button – це кнопка, призначена для опрацювання подій, тобто у разі її натискання використовується певна запрограмована подія.

Для правильного керування програмою реалізовано три головні кнопки:

· “З’єднатись” – встановлює зв’язок між головною програмою і програмою “Сервером”;

· “Передати” – кнопка для передавання завантаженого зображення на сервер;

· “Завантажити знак” – кнопка для завантаження поточного зображення у вибране користувачем місце.

2. Timer – елемент керування, який використовується для виконання дії через певний проміжок часу.

Основна властивість елемента – задає інтервал часу, що визначає, як часто таймер повинен повідомляти застосунок. Властивість Interval задають в мілісекундах (1000 мілісекунд = 1 секунда).

Зауважимо, що максимально можливе значення цієї властивості дорівнює 65 секундам.

3. SaveFileDialog – це клас, за допомогою якого відбувається збереження потрібних користувачеві файлів. Він має такі головні властивості:

· DefaultExt: встановлює розширення файла, яке додається за замовчуванням, якщо користувач увів ім’я файла без розширення;

· AddExtension: за значення true додає до імені файла розширення. Розширення беруть з властивості DefaultExt або Filter;

· CheckFileExists: якщо має значення true, то перевіряє наявність файла із вказаним ім’ям;

· CheckPathExists: якщо має значення true, то перевіряє існування шляху до файла з вказаним ім’ям;

· FileName: повертає повне ім’я файла, вибраного в діалоговому вікні;

· Filter: задає фільтр файлів, завдяки чому в діалоговому вікні можна відфільтрувати файли з розширенням;

· InitialDirectory: встановлює каталог, який відображається під час першого виклику вікна;

· Title: заголовок діалогового вікна;

4. MainMenu – елемент є своєрідним контейнером для окремих пунктів меню, представлених об’єктом ToolStripMenuItem.

Найважливіші властивості компонента MenuStrip:

· Dock: прикріплює меню до однієї зі сторін форми;

· LayoutStyle: задає орієнтацію панелі меню на формі. Може набувати таких значень:

a. HorizontalStackWithOverflow: розташування по горизонталі з переповненням – якщо довжина меню перевищує довжину контейнера, то нові елементи, що виходять за межі контейнера, не відображаються, тобто панель переповнюється елементами;

b. StackWithOverflow: елементи розташовуються автоматично з переповненням;

c. VerticalStackWithOverflow: елементи розміщуються вертикально з переповненням;

· ShowItemToolTips: вказує, чи відображатимуться підказки, які спливають, для окремих елементів меню;

· Stretch: дає змогу розтягнути панель по всій довжині контейнера;

· TextDirection: задає напрямок тексту в пунктах меню.

5. ComboBox – елемент керування, використовуваний для відображення даних у випадному полі зі списком. Елемент використовується в ситуаціях:

· коли користувачеві необхідно вибрати одне або кілька значень зі списку розміром від чотирьох до кількох десятків позицій. Якщо позицій менше, то простіше застосувати перемикачі, якщо більше, то орієнтуватися в списку стає незручно і необхідно використовувати спеціальні прийоми, коли користувач вводить перші літери потрібного слова і в списку залишаються тільки значення, які починаються на ці букви;

· коли список позицій для вибору необхідно формувати динамічно на підставі даних з джерела (бази даних, листа Excel тощо).

Логічні моделі даних подають абстрактну структуру області інформації. Для подання логічної структури програми необхідно здійснити опис усіх основних функцій системи.

Програма містить такі функції:

1. Private void buttonX1_Click(object sender, EventArgs e) – кнопки для функції, що забезпечує опрацювання подій і передає завантажене зображення на сервер.

2. Private void buttonX5_Click(object sender, EventArgs e) – кнопки для функції, що забезпечує опрацювання подій, яка встановлює з’єднання між головною формою та сервером.

3. Private void sideNavItem8_Click(object sender, EventArgs e) – кнопки для функції, що забезпечує опрацювання подій і дає змогу зберегти поточний результат роботи програми.

4. Private void CropImage(Rectangle section) – функція з параметром, яка обрізає потрібний сектор зображення для коректного збереження і передавання даних.

5. Private void ChangeVidst(int dr, int vor, int neyt, int nevid) – функція, яка задає відстань між різними елементами зображення.

6. Private void ChangeImg() – зберігає в собі змінене зображення.

7. Private void Modify1() – функція для подання додаткової інформації про символ у першому секторі.

8. Private void Modify2() – функція, для подання додаткової інформації про символ у другому секторі.

9. Private void StockImg() – зберігає символи без змін, у початковому форматі.

10. Private void sideNavItem5_Click(object sender, EventArgs e) – функція для переходу програми в компактний стиль і навпаки.

11. Private void DrawString_right(Graphics g, string text, PointF location) – малює праву частину ампліфікаторів символу.

12. Private void DrawString_left(Graphics g, string text, PointF location) – малює ліву частину ампліфікаторів символу.

13. Private void pictureBox2_MouseDown(object sender, MouseEventArgs e) – функція для малювання графічного ампліфікатора швидкості руху.

Для реалізації та функціонування програми необхідне таке програмне забезпечення:

- операційна система Microsoft Windows 7,10;
- середовище програмування Visual Studio 2015.

Мінімальні характеристики для використання програми:

- частота процесора – 1000 MHz;
- обсяг оперативної пам’яті – 256 MB;
- обсяг жорсткого диска – 80 GB;
- обсяг пам’яті графічного адаптера – 128 MB.

Для запуску програми необхідно запустити файл “Symbol.exe”, після цього з’являється завантажувальне вікно (рис. 18), а після завантаження програми головне вікно генератора (рис. 19).

Рис. 18. Завантажувальне вікно програми

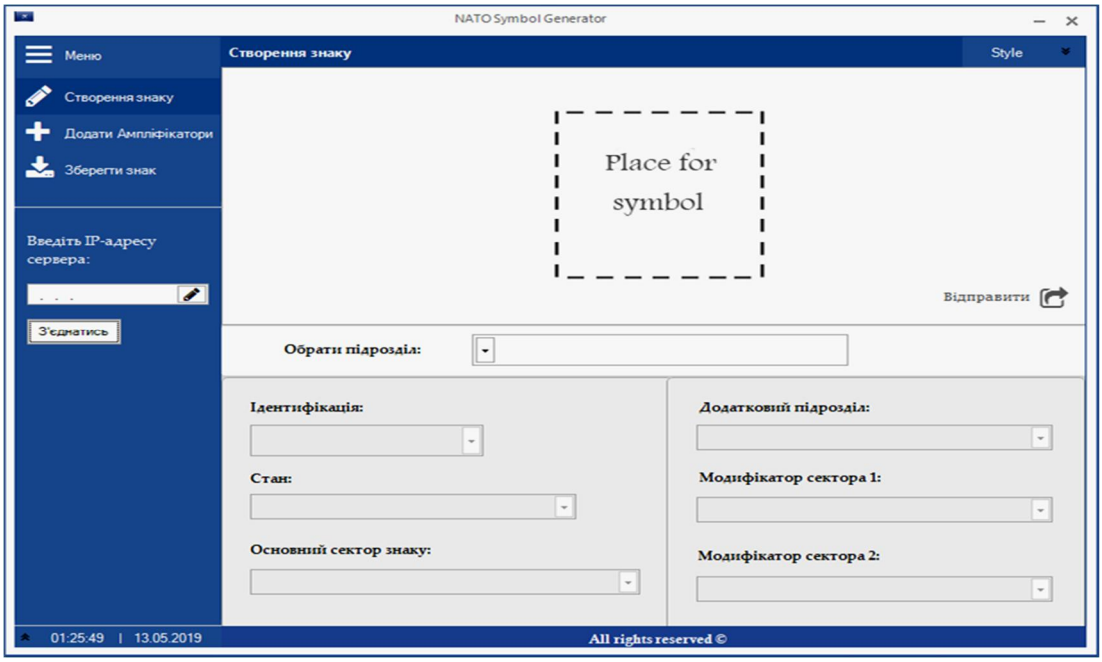


Рис. 19. Головне вікно програми

Після того, як з’явилося головне вікно програми, користувачеві надається можливість вибрати підрозділ, відповідно до якого формуватиметься умовний знак. Потрібно натиснути на поле з випадним списком напроти надпису “Обрати підрозділ”. З’явиться список, з якого можна вибрати потрібний підрозділ (рис. 20).

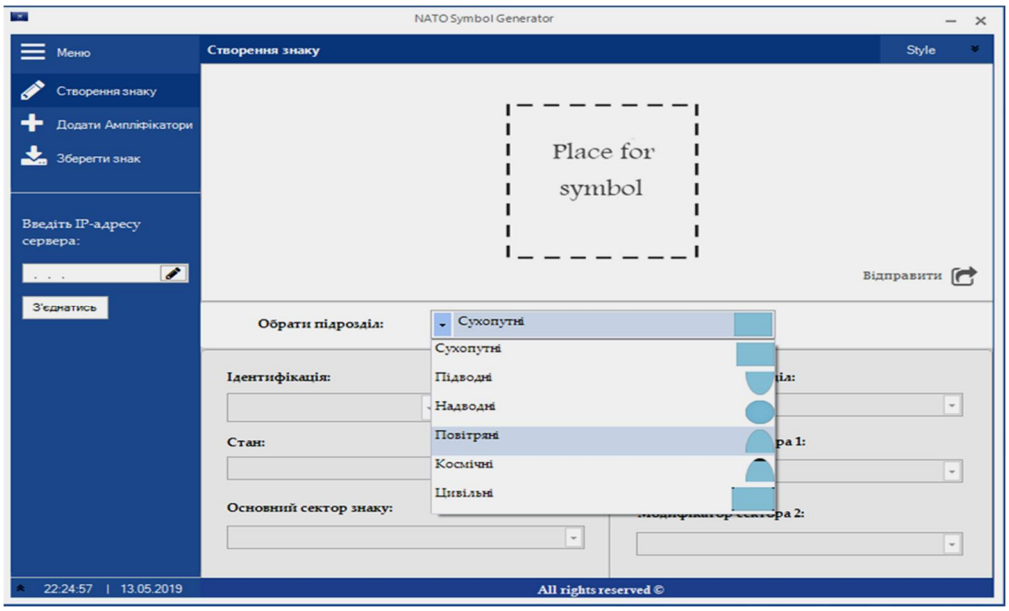


Рис. 20. Список військових підрозділів

Після того, як вибрано підрозділ, на робочій області з'являється відповідний умовний знак. За замовчуванням знак зображають з ідентифікацією "Дружні". Відкриваються такі можливості, як вибрати Ідентифікацію та Стан для вибраного підрозділу (рис. 21). Тепер можна зберігати знак або, перейшовши на вкладку "Додати ампліфікатори", доповнити додатковою інформацією про нього.

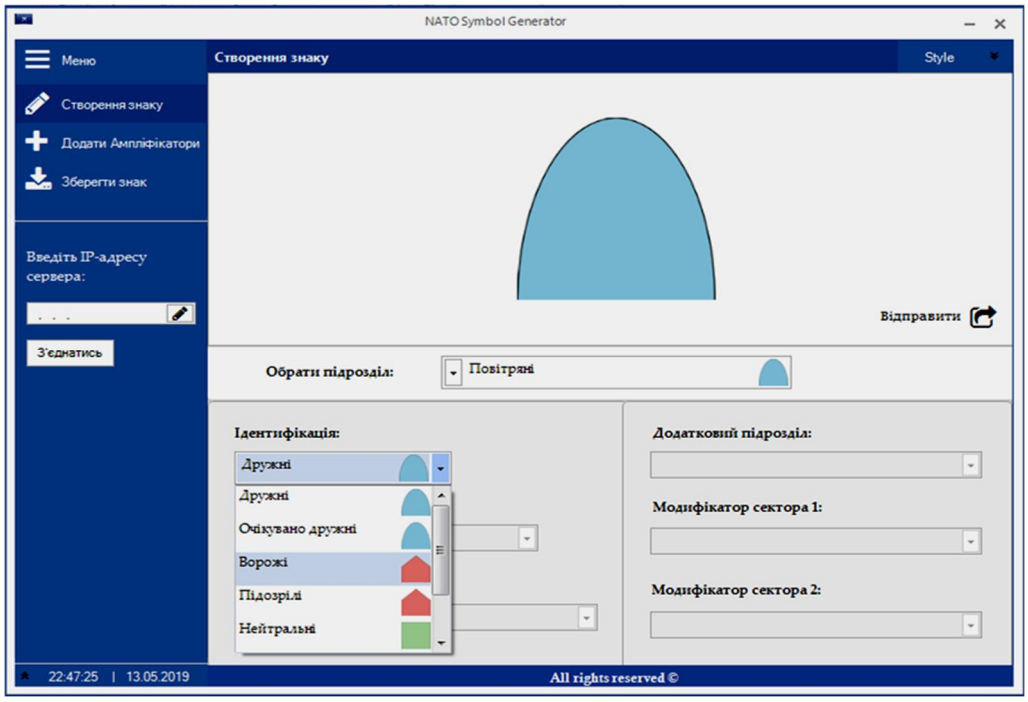


Рис. 21. Ідентифікація військових підрозділів

Коли користувач почне змінювати Ідентифікацію військових підрозділів, програма дозволить вибрати основний сектор знака. Основний сектор повинен міститись у центрі зображення, відповідно до стандартизації умовних знаків. Для цього потрібно натиснути на поле з випадним списком, розташоване під надписом "Основний сектор знака". Випаде список, який складатиметься з елементів основних позначок (рис. 22).

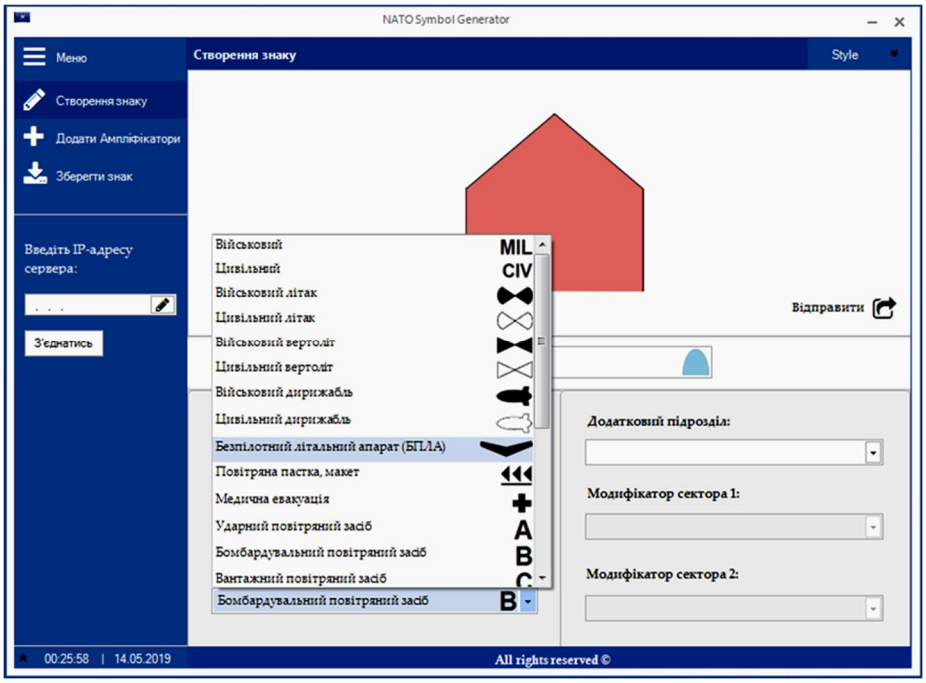


Рис. 22. Вибір позначки для основного сектора

Після того, як вибрано основну позначку, можна наносити на зображення додаткову інформацію про знак у вигляді модифікаторів.

Модифікатори бувають двох типів:

· модифікатор першого сектора (міститься у верхній частині знака);

· модифікатор другого сектора (у нижній частині відносно основної позначки).

Додавання модифікаторів здійснюється аналогічно попереднім діям.

Потрібно натиснути на поля випадних списків, місце розташування яких в лівій частині вікна під надписами відповідно: “Модифікатор сектора 1” і “ Модифікатор сектора 2”

Приклад вибору та додавання модифікаторів першого сектора наведено на рис. 23.

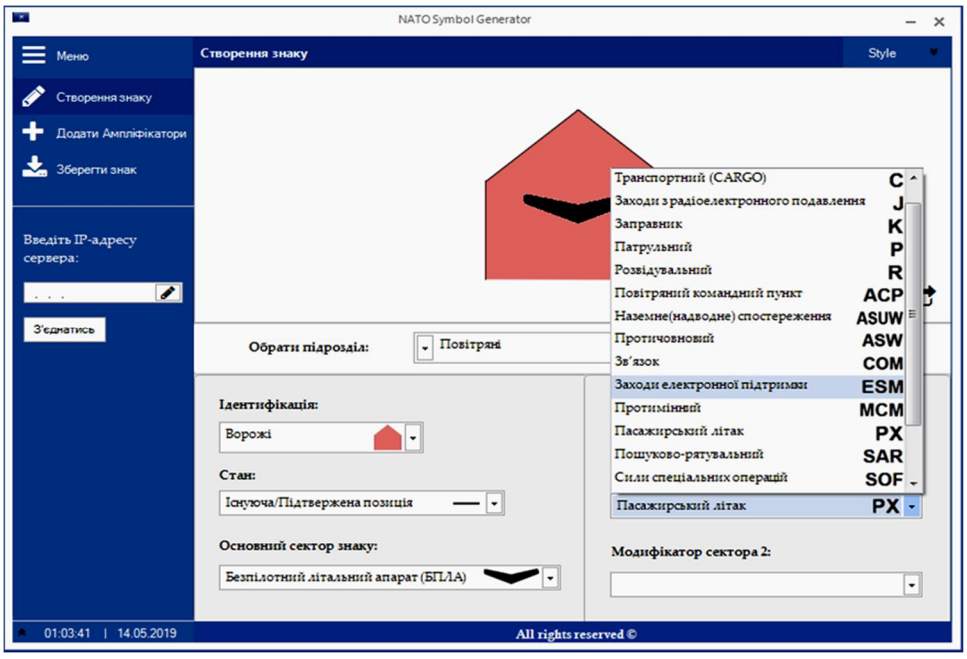


Рис. 23. Додавання модифікатора першого сектора

Приклад вибору та додавання модифікаторів другого сектора наведено на рис. 24.

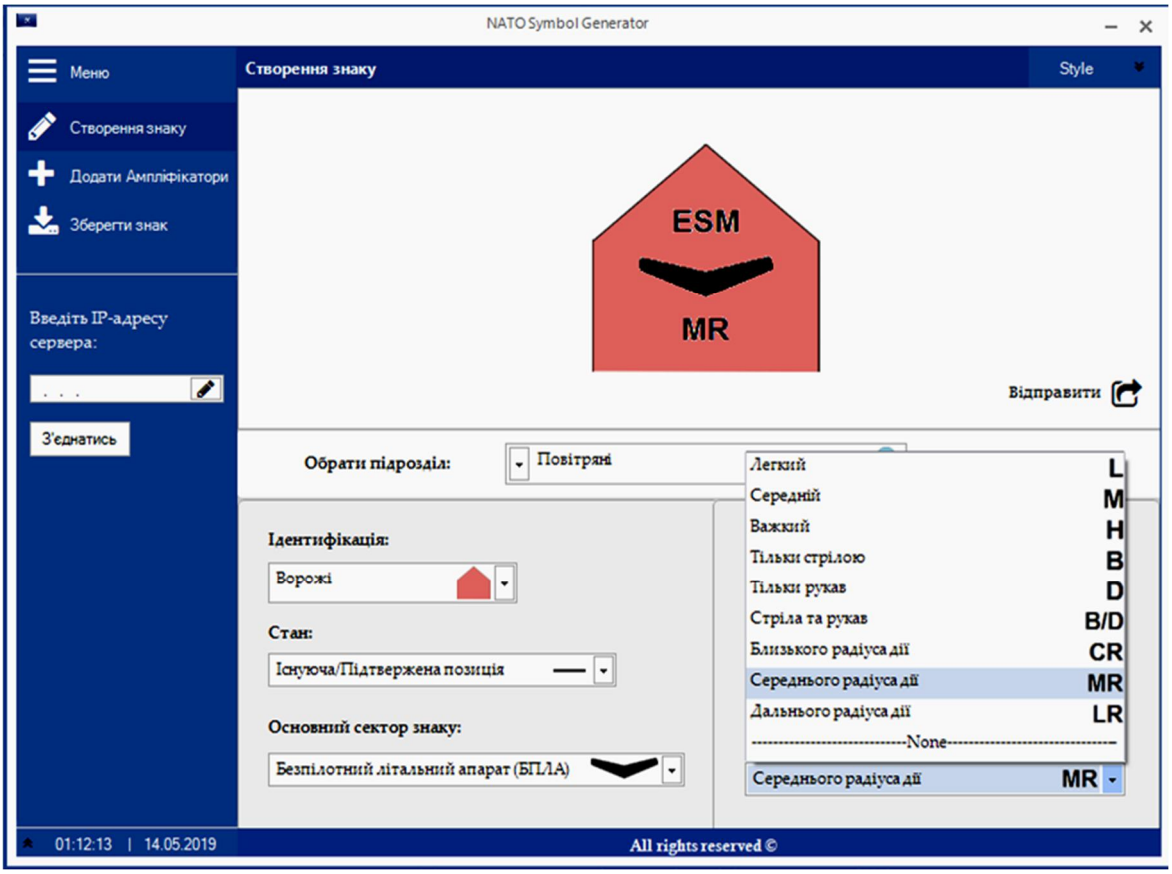


Рис. 24. Додавання модифікатора другого сектора

Для деяких підрозділів, а конкретно для сухопутних та повітряних підрозділів, можна вибирати додаткові підрозділи. Якщо натиснути на поле під надписом “Додатковий підрозділ”, можна вибрати потрібний. Деякі можливості програми будуть вимкнені, оскільки не для всіх умовних знаків існують модифікатори, або доступ до зміни основного сектора знака буде неможливий (рис 25).

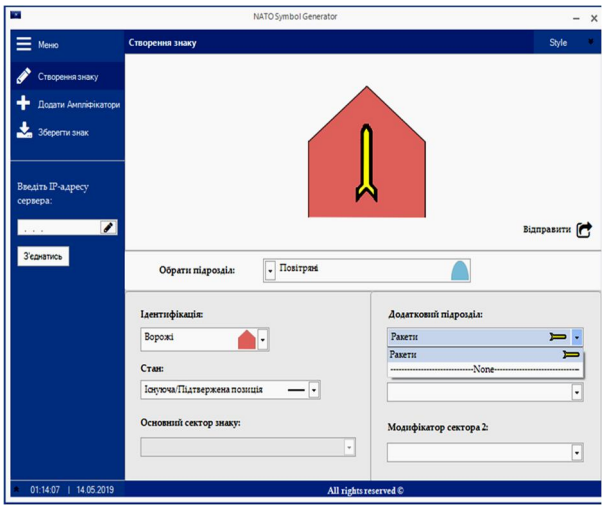


Рис. 25. Підрозділ повітряних знаків “Ракети”

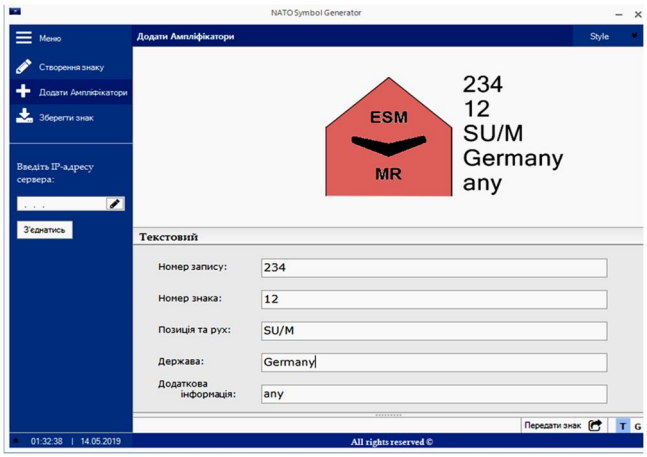


Рис. 26. Вкладка “Додати Ампліфікатори”

Виконавши усі необхідні дії для створення основної частини знака, можна перейти до внесення додаткової інформації про поточний символ. Для цього потрібно перейти з вкладки “Створення знака” на вкладку “Додавання Ампліфікаторів”. Ампліфікатори бувають двох видів: текстові та графічні. За замовчуванням на вкладці спочатку викликається вкладка з текстовими ампліфікаторами. Для додавання інформації на зображення потрібно у відповідні поля ввести дані про символ (рис. 26). Для кожного типу підрозділу кількість і типи додаткової інформації різні. Якщо вибрати підрозділ “Сухопутні”, то кількість текстових ампліфікаторів збільшиться вдвічі (рис. 27).

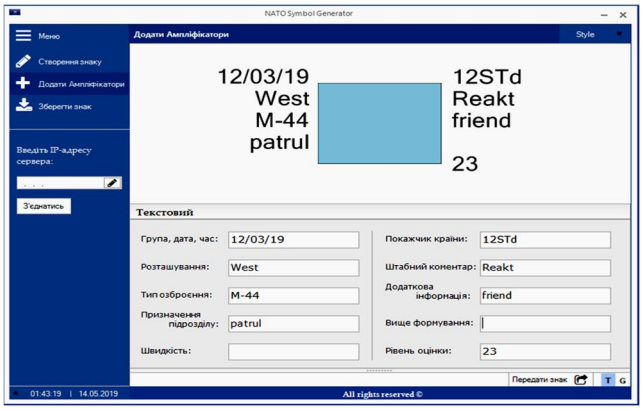


Рис. 27. Ампліфікатори Сухопутних військ

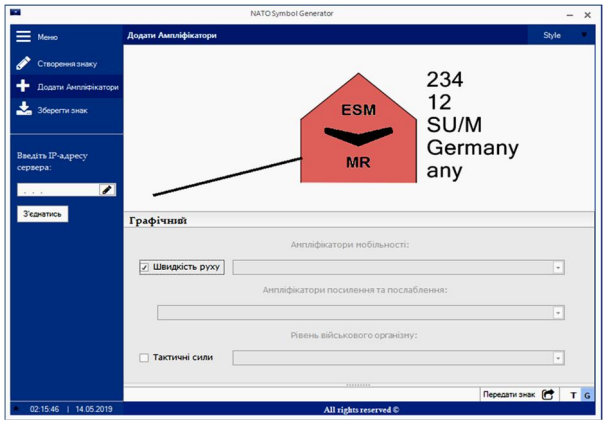


Рис. 28. Графічний ампліфікатор швидкості

Як вказано вище, окрім текстових ампліфікаторів, існують також графічні. Але доступ до деяких типів поля для додавання графічної інформації може також бути обмежений відповідно до вибраного підрозділу з попередньої вкладки. Наприклад, можна додати ампліфікатор швидкості. Для додавання потрібно перейти на другу вкладку із буквою “G”. Поставити позначку напроти надпису “Швидкість руху” та натисканням правою кнопкою миші на робочу область малювати лінію швидкості, яка визначатиме і напрям об’єкта (рис. 28).

Після того як умовний знак повністю створено, його можна зберегти, натиснувши кнопку “Зберегти знак”. Для коректного використання збереженого зображення воно автоматично буде обрізано відповідно до кількості введеної текстової інформації.

Зображення можна також передавати на сервер. Це окремий застосунок, який потрібно запустити на комп’ютері, що прийматиме зображення. Якщо відповідний застосунок не запущений, видаватиметься повідомлення про помилку (рис. 29).

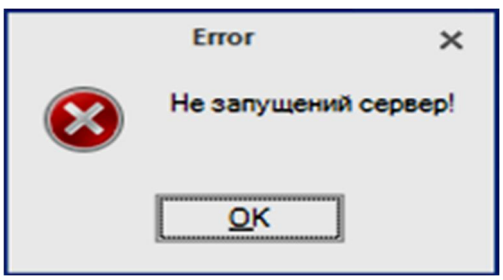


Рис. 29. Повідомлення про помилку

Рис. 30. Головне вікно програми “Server”

Для запуску застосунку необхідно запустити файл “Server.exe”, після цього з’явиться головне вікно програми (рис. 30). Для початку приймання зображень потрібно натиснути на кнопку “Запуск

сервера”, після натискання якої на робочу область буде виведено повідомлення про очікування підключення (рис. 31).

Рис. 31. Очікування підключення

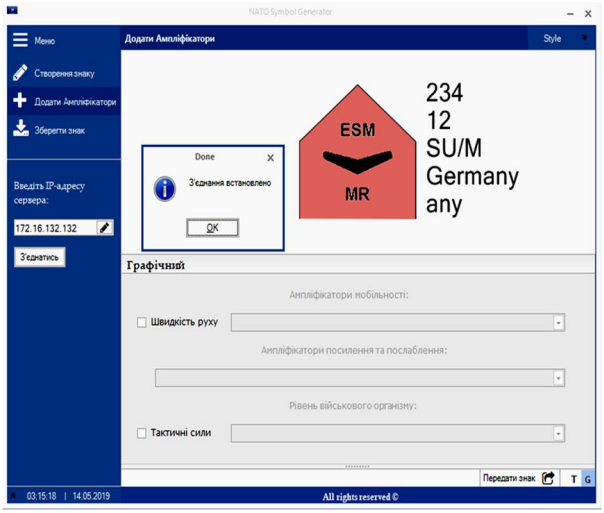


Рис. 32. Очікування підключення

Під робочою областю для приймання зображень розміщене статичне поле. Після запуску інформаційної системи у це поле записується локальна IP-адреса машини, з якої запущено сервер.

У клієнтському застосунку “Symbol”, в лівій частині програми, є поле для введення IP-адреси комп’ютера, до якого потрібно підключитись. Для передавання зображень між клієнтом та сервером у це поле потрібно ввести IP-адресу комп’ютера, на якому запущено застосунок з сервером, та натиснути на кнопку “З’єднатись”. В результаті як у клієнтському застосунку, так і на сервері з’явиться повідомлення про те, що з’єднання встановлено (рис 32). Як тільки з’єднання буде встановлено, можна починати передавати будь-які зображення, створені програмою. Після успішного передавання зображення на сервер застосунок відповідно відреагує повідомленням про успішне передавання, таке саме повідомлення буде відображено й на сервері (рис. 33, 34).

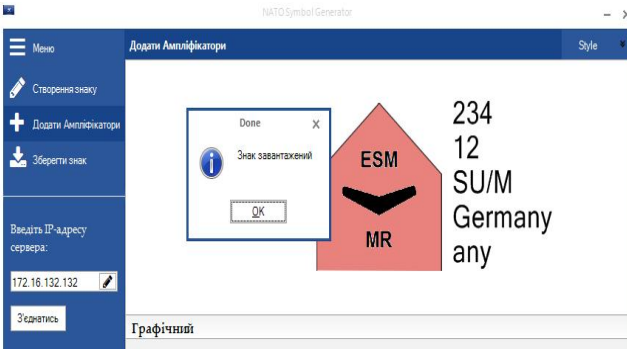


Рис. 33. Повідомлення для клієнтської програми

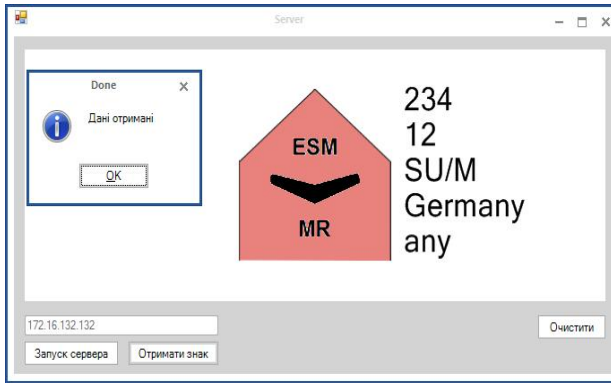


Рис. 34. Результат передавання зображення

Якщо не подобається колір або стиль кнопок та панелей генератора, це легко виправити за допомогою панелі, яка вилітає, у верхньому правому кутку. Біля вибраного стилю візуального кольору програми автоматично встановлюється позначка, а в нижньому правому куту міститься невеликий органайзер, на якому можно подивитись поточний час та дату (рис. 35).

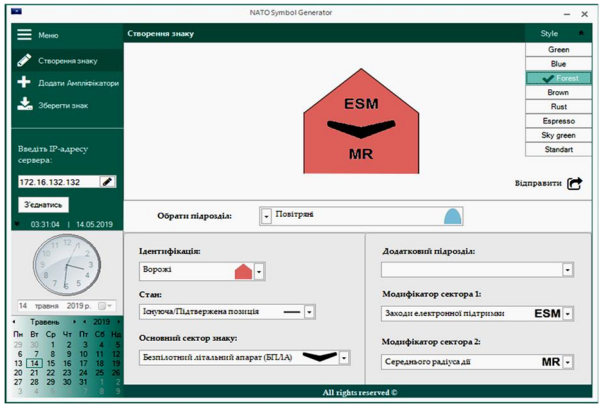


Рис. 35. Змінений стиль вікна

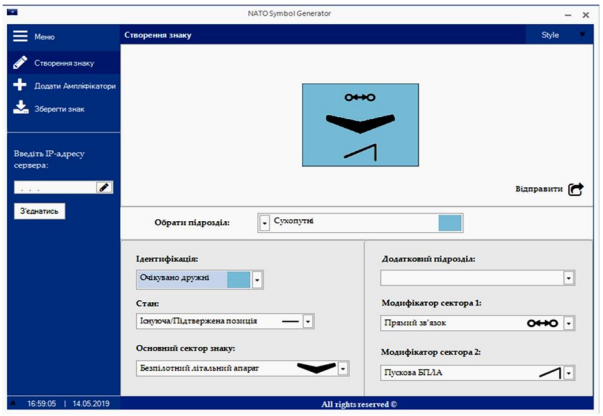


Рис. 36. Вкладка "Створення знака"

Щоб перевірити правильність виконання програми, створимо умовний військовий знак Сухопутного підрозділу. Після цього виконаємо усі необхідні дії для внесення додаткової інформації про символ на вкладці "Створення знака", яка відкриється за замовчуванням (рис. 36). Для внесення додаткової текстової інформації про символ необхідно перейти на вкладку "Додати Ампліфікатори" і в спеціально відведені для цього поля ввести тестову інформацію. Вся введена інформація буде динамічно відображатися. Коли символ створено, виконують всі необхідні дії для встановлення зв'язку між двома машинами. Для цього потрібно запустити програму на сервер та натиснути на кнопку "Запустити сервер". У клієнт-програмі необхідно ввести IP-адресу сервера та натиснути кнопку "З'єднатись". Після цього можна передавати знак. Створивши основну частину знака, додамо текстову інформацію на вкладці "Додати Ампліфікатори" (рис. 37).

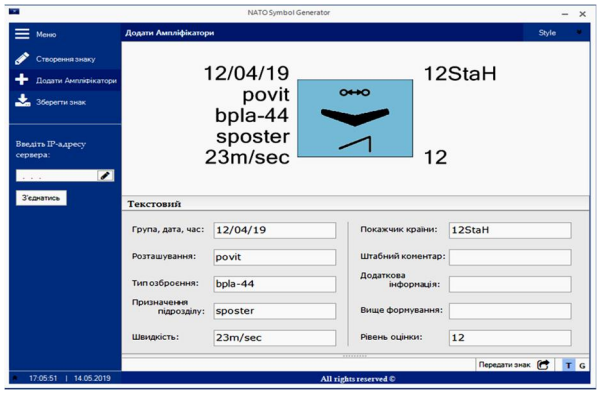


Рис. 37. Вкладка "Додати Ампліфікатори"

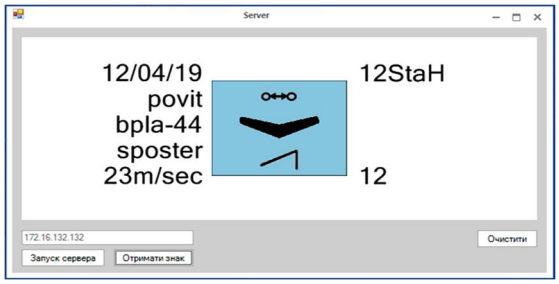


Рис. 38. Результат передавання зображення на сервер

Після з'єднання програм між собою необхідно натиснути кнопку "Відправити", в результаті зображення з'явиться на сервері (рис. 38).

Висновки

Розроблення програми, яка використовуватиметься як генератор умовних тактичних знаків у стандарті APP-6D; незважаючи на простоту в користуванні, супроводжується певними ускладненнями, для подолання яких використано усі основні прийоми мови програмування C#. Програмування в середовищі візуальних систем на мові C# можна розділити на кілька рівнів. Перший – розроблення прикладної програми. В цьому випадку, створюючи програму з графічним інтерфейсом користувача, як правило, використовували лише готові компоненти з уже запрограмованими властивостями.

Зовнішня відмінність між властивостями і полями в цьому випадку не дуже заважає, оскільки йдеться про використання уже налагоджених бібліотек, а в розпорядженні є вбудована система допомоги. І навіть більше, чимало властивостей візуальних компонентів навіть не фігурують в тій частині програми, яку потрібно програмувати вручну. Значення властивостей вікон, кнопок і інших елементів інтерфейсу задані в режимі діалогу з візуальною системою. У візуальному середовищі, по суті, потрібно мати справу не з усією мовою програмування, а лише з тією її частиною, яка передбачає можливість використання готових властивостей готових класів. Другий рівень – розроблення візуальних компонентів. У цьому випадку створена власна система класів. У цій ситуації, коли властивості застосовуються не тільки для візуальних елементів, відсутність відмінності між викликом підпрограм (get і set) від звернення до поля може погіршити можливості розуміння програми. За текстом програми вже не можна зрозуміти, чи зводиться дія до зміни або до отримання значення поля, чи воно пов'язане з виконанням інших операцій. Результатом цього дослідження є систематизація, закріплення та розширення теоретичних і практичних знань стосовно створення програмного забезпечення для генерації умовних знаків. Результатом застосування цих знань є побудова інтелектуальної інформаційної системи автоматичної генерації умовних знаків у стандарті APP-6D, яка на момент написання статті не має десктопних аналогів.

Список літератури

1. Venda V. (1995). Ergodynamics: theory and applications. *Ergonomics*, 38(8), 1600–1616. DOI: 10.1080/00140139508925212

2. Buch G., Rambo D., & Jacobson I. (2006). UML. SPb: Piter.

3. Troelsen A. (2003). C# and the. NET Platform. Apress, DOI: 10.1007/978-1-4302-1141-9.

4. Joos G. (2022). Geographic Information Systems in Defense, Springer Handbook of Geographic Information. Cham: Springer International Publishing, 685–705. DOI: 10.1007/978-3-030-53125-6_25.

5. Tanjimuddin M., Kannisto P., Jafary P., Filppula M., Repo S., Hästbacka D. (2022). A comparative study on multi-agent and service-oriented microgrid automation systems from energy internet perspective. *Sustainable Energy, Grids and Networks*, 32, 100856. DOI: 10.1016/j.segan.2022.100856.

6. West J. (2022). Data Communication and Computer Networks: A Business User's Approach. Cengage Learning.

7. Akhtar A., Bakhtawar B., Akhtar S. (2022). Extreme Programming Vs Scrum: A Comparison Of Agile Models. *International Journal of Technology, Innovation and Management* (IJTIM), 2(2). DOI: 10.54489/ijtim.v2i2.77

8. Budd T. (2008). Introduction to object-oriented programming. Pearson Education India.

9. Gamma E., Helm R., Johnson R., Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. In ECOOP'93 – Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings, 7, 406–431. Springer Berlin Heidelberg. DOI: 10.1007/3-540-47910-4_21.

10. Holland I. M., Lieberherr K. J. (1996). Object-oriented design. *ACM Computing Surveys*, 28(1), 273–275.

11. Bierman G. M., Meijer E., Torgersen M. (2007). Lost in translation: formalizing proposed extensions to C#. In Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications October, 479–498. DOI: 10.1145/1297027.1297063.

12. Kotler, Philip (1967). Marketing Management: Analysis, Planning and Control. Englewood Cliffs, N. J.: Prentice-Hall.

13. Згуровський М. З., Панкратова Н. Д. (2007). Основи системного аналізу. К.: BHV.

14. Досин Д. Г., Литвин В. В., Нікольський Ю. В., Пасічник В. В. (2009). Інтелектуальні системи, базовані на онтологіях. Львів: Вид. дім "Цивілізація".

15. Technical news. Ukraine Eng. association in Lviv, 2009.

16. Carmichael P. (2003). Teachers as researchers and teachers as software developers: how use-case analysis helps build better educational software. *The Curriculum Journal*, 14(1), 105–122. DOI: 10.1080/0958517032000055983

17. Литвин В. В., Пасічник В. В., Яцишин Ю. В. (2009). Інтелектуальні системи. Львів: Новий світ-2000.

18. Shin Y. C., Xu C. (2017). Intelligent systems: modeling, optimization, and control. CRC press.

19. Moulin H. (1991). Axioms of cooperative decision making (No. 15). Cambridge university press

20. Garcia-Molina H., Ullman J. D., Widom J. (2000). Database system implementation, Vol. 672. Upper Saddle River: Prentice Hall, 2000. URL: https://www.csd.uoc.gr/~hy460/pdf/000.pdf

21. Awange J., Kiema J. B. (2013). Environmental geoinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg, 10, 978–3. DOI: 10.1007/978-3-030-03017-9.

22. Bishop J. (2007). C# 3.0 Design Patterns: Use the Power of C# 3.0 to Solve Real-World Problems. O'Reilly Media.

23. Bishop J., Horspool R. N., Worrall B. (2005). Experience in integrating Java with C# and. NET. *Concurrency and Computation: Practice and Experience*, 17(5–6), 663–680. DOI: 10.1002/cpe.858.

24. Fülö J. (2005). Introduction to decision making methods. In BDEI-3 workshop, Washington, 1–15.

25. Sumathi S., Esakkirajan S. (2007). Fundamentals of relational database management systems. Springer.

References

1. Venda V. (1995). Ergodynamics: theory and applications. *Ergonomics*, 38(8) 1600–1616. DOI: 10.1080/00140139508925212

2. Buch, G., Rambo, D., & Jacobson, I. (2006). UML. SPb : Piter.

3. Troelsen A. (2003). C# and the. NET Platform. Apress, DOI: 10.1007/978-1-4302-1141-9.

4. Joos G. (2022). Geographic Information Systems in Defense, Springer Handbook of Geographic Information. Cham: Springer International Publishing, 685–705. DOI: 10.1007/978-3-030-53125-6_25.

5. Tanjimuddin M., Kannisto P., Jafary P., Filppula M., Repo S., Hästbacka D. (2022). A comparative study on multi-agent and service-oriented microgrid automation systems from energy internet perspective. *Sustainable Energy, Grids and Networks*, 32, 100856. DOI: 10.1016/j.segan.2022.100856.

6. West J. (2022). Data Communication and Computer Networks: A Business User's Approach. Cengage Learning.

7. Akhtar A., Bakhtawar B., Akhtar S. (2022). Extreme Programming Vs Scrum: A Comparison Of Agile Models. *International Journal of Technology, Innovation and Management (IJTIM)*, 2(2). DOI: 10.54489/ijtim.v2i2.77.

8. Budd T. (2008). Introduction to object-oriented programming. Pearson Education India.

9. Gamma E., Helm R., Johnson R., Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. In ECOOP'93 – Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings, 7, 406–431. Springer Berlin Heidelberg. DOI: 10.1007/3-540-47910-4_21.

10. Holland I. M., Lieberherr K. J. (1996). Object-oriented design. *ACM Computing Surveys*, 28(1), 273–275.

11. Bierman G. M., Meijer E., Torgersen M. (2007). Lost in translation: formalizing proposed extensions to C#. In Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications, 2007, October, 479–498. DOI: 10.1145/1297027.1297063.

12. Kotler, Philip (1967). Marketing Management: Analysis, Planning and Control. Englewood Cliffs, N. J.: Prentice-Hall.

13. Zgurovsky M. Z., Pankratova N. D. (2007). Fundamentals of system analysis. K.: BHV.

14. Dosyn D. H., Lytvyn V. V., Nikol′s′kyy Yu. V., Pasichnyk V. V. (2009). Intelektual′ni systemy, bazovani na ontolohiyakh. L′viv : Vyd. dim "Tsyvilizatsiya".

15. Technical news. Ukraine Eng. association in Lviv, 2009.

16. Carmichael P. (2003). Teachers as researchers and teachers as software developers: how use-case analysis helps build better educational software. *The Curriculum Journal*, 14(1), 105–122. DOI: 10.1080/0958517032000055983.

17. Lytvyn V. V., Pasichnyk V. V., Yatsyshyn Yu. V. (2009). Intellectual systems. Lviv: New world-2000.

18. Shin Y. C., Xu C. (2017). Intelligent systems: modeling, optimization, and control. CRC press.

19. Moulin H. (1991). Axioms of cooperative decision making, No. 15. Cambridge university press.

20. Garcia-Molina H., Ullman J. D., Widom J. (2000). Database system implementation, ol. 672). Upper Saddle River: Prentice Hall URL: https://www.csd.uoc.gr/~hy460/pdf/000.pdf

21. Awange J., Kiema J. B. (2013). Environmental geoinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg, 10, 978-3. DOI: 10.1007/978-3-030-03017-9

22. Bishop J. (2007). C# 3.0 Design Patterns: Use the Power of C# 3.0 to Solve Real-World Problems. O'Reilly Media,

23. Bishop J., Horspool R. N., Worrall B. (2005). Experience in integrating Java with C# and. NET. Concurrency and Computation: Practice and Experience, 17(5-6), 663–680. DOI: 10.1002/cpe.858

24. Fülö J. (2005). Introduction to decision making methods. In BDEI-3 workshop, Washington, 1–15.

25. Sumathi S., Esakkirajan S. (2007). Fundamentals of relational database management systems, 47, Springer.

INTELLIGENT INFORMATION SYSTEM FOR AUTOMATIC GENERATION OF SYMBOLS IN THE APP-6D STANDARD

Vasyl Lytvyn [1], Anton Rynkov [1], Victoria Vysotska[1,2]

[1] Lviv Polytechnic National University,
Information Systems and Networks Department, Lviv, Ukraine
[2] Osnabrück University, Institute of Computer Science, Osnabrück, Germany
E-mail: Vasyl.V.Lytvyn@lpnu.ua, ORCID: 0000-0002-9676-0180
E-mail: Anton.Rynkov.mnsa.2019@lpnu.ua, ORCID: 0000-0002-5751-9052
E-mail: Victoria.A.Vysotska@lpnu.ua, ORCID: 0000-0001-6417-3689

Military symbols play a key role in the command and control of military forces. By communicating information that meets basic requirements, situational awareness can be quickly achieved; by its graphical nature, it provides a common operating language that greatly facilitates interaction across cultural and linguistic barriers. With the advent of information technology, the need arose for rapidly recognized international standards that could then be taught to computers. The fusion of high standard air, sea and land symbols on APP-6 paper resulted in Mil-Std-2525 and NATO APP-6D. This standard applies to all NATO ground components that are directly or partially involved in C4I operations, system operations, systems development and training in the context of NATO ground component operations. When designing and developing signs, composing their components, factors such as the ability to easily recognize signs, their legibility in different lighting conditions on a diverse cartographic basis, in different sizes and types of electronic screens with different degrees of physical and intellectual fatigue should be taken into account. The conventional signs standard concentrates signs for use in modern multi-chromatic electronic systems. At the same time, all signs can be used in monochrome systems and for freehand decoration. Development of this software will allow creating comfortable conditions for performing such actions as: create and edit tactical graphics of military symbols; ensuring compliance with all standards regarding the conditions of standardization of APP-6D; performing secure transmission of characters between multiple users. The object of the research is an automatic generator of conventional symbols in the tactical military standard APP-6D. The subject of the research is software for automatic generation of tactical symbols in the APP-6D standard. The aim of the work is to develop software that will automatically generate tactical insignia in the APP-6D standard. A comparative analysis of various methods for the development of such systems for generating symbols has been carried out. The project was being developed and it does not have a single equivalent in the Ukrainian language. Also, there are not a single desktop implementation. The project can be part of the software used by the Ukrainian military.

Key words: automatic generator; APP-6D standard; software; graphics; symbol; sign; information system.