НУ “Львівська політехніка”

На правах рукопису

Прізвище Автора Роботи

УДК 004.853+004.855.5

Це стаб для оформлення дисертації, курсової, магістерської

01.05.03 — математичне та програмне забезпечення обчислювальних машин і систем

Курсова робота

Науковий керівник
Прізвище Керівника Роботи,
кандидат фізико-математичних наук, доцент

Львів — 2012

ЗМІСТ

ВСТУП

Актуальність теми. У повсякденному житті кожен із нас часто стикається з різноманітною інформацією. Важливими аспектами її подання є те наскільки вона буде зрозумілою, скільки людей зацікавляться нею і як добре її засвоять.

Невід'ємним елементом опрацювання інформації є комп'ютер. Він може надзвичайно ефективно і точно виконувати ці завдання які через велику масштабність чи складність є неможливими для людського мозку. Дисципліна яка вивчає взаємодію людини з комп'ютером, оцінку та реалізацію інтерактивних обчислювальних систем та явищ, а також те яким чином комп'ютерні технології впливають на людську працю та розвиток, це — Human-Computer Interaction (HCI) [17].

Складовою HCI є візуалізація даних. Це важливий і потрібний процес, оскільки неструктуровані дані які подані за допомогою символів є важкими для сприйняття людиною. Розділом HCI який займається візуалізацією є інфографіка.

Інформаційна графіка або інфографіка — це розробка та вивчення графічного візуального подання інформації, даних і знань, створеного з метою представлення складної інформації швидко і чітко [9].

Інфографіка має важливе місце у всіх сферах нашого життя, зокрема: політичній, освітній та інших. Великою популярності набула візуалізація даних і в ЗМІ, що дає змогу легше донести інформацію до людей, подати її у зрозумілій формі та підняти рівень зацікавлення нею населення.

У системному аналізі, для кращої й ефективнішої роботи, часто використовується візуалізація. Неструктурована інформація потребує додаткового опрацювання для можливості ефективної роботи з нею. Саме тому

системні аналітики використовують інфографіку, як засіб особливої підготовки початкових даних з якими буде проводитись подальша робота та на основі яких, після проведення аналізу, прийматимуть рішення.

Дана тема є актуальною у наш час і потребує додаткового дослідження й розробки нових методів та засобів, оскільки з рівнем інформатизації суспільства зростає зацікавленість у якості подання та розуміння даних які нас оточують.

Метою роботи є візуалізація радарів авіокомпанії на карті з можливістю інтерактивного відображення їхнього технічого стану й території яку вони покривають. Для досягнення цієї мети були сформульовані та вирішені такі основні завдання:

— розробити методи та алгоритми для досягнення мети;

— знайти і освоїти необхідні засоби для реалізації;

— розробити програмне забезпечення.

Об'єктом дослідження є процес інтерактивного візуального відображення даних у зручній для сприйняття формі.

Предметом дослідження є створення підходів до передачі абстрактної інформації в легку для сприйняття форму.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ТА ІНШИХ ДЖЕРЕЛ

1.1. Загальний огляд

Сучасним дослідженням візуалізації передувала комп'ютерна графіка, яка з самого початку використовувалась для вивчення наукових проблем. Тим не менше, у перші дні відсутності графічної потужності найчастіше обмежується її корисність. Останній акцент на візуалізації був поставленій в 1987 році внаслідок особливого питання щодо комп'ютерної графіки на візуалізації в наукових обчисленнях. Відтоді було проведено кілька конференцій та семінарів. Вони були присвячені загальним темам візуалізації даних, візуалізації інформації і наукової візуалізації, і більш конкретних областям, таким, як обсяг візуалізації.

Галузь візуалізації інформації з'явилася внаслідок досліджень взаємодії людини і комп'ютера, комп'ютерних наук, графіків, дизайну, психології та бізнес-методів. Вона все частіше застосовується в якості найважливішого компонента в наукових дослідженнях, цифрових бібліотеках, інтелектуальному аналізі даних, аналізі фінансових даних, дослідженнях ринку, виробничого контролю, і дослідженні ліків. Візуалізація інформації припускає, що візуальні уявлення і методи взаємодії користуються здатністю людського ока пропускати інформацію в мозок, щоб користувачі могли побачити, вивчити і зрозуміти велику кількість інформації за один раз. Візуалізація інформації спрямована на створення підходів до передачі абстрактної інформації в інтуїтивно зрозумілі способи.

Аналіз даних є невід'ємною частиною всіх прикладних досліджень та

вирішення проблем в промисловості. Найбільш фундаментальні підходи аналізу даних — візуалізація, статистика, видобуток даних і методи машинного навчання. Серед усіх цих підходів, візуалізації інформації, або, візуального аналізу даних, є той, який спирається в основному на пізнавальні навички аналітиків, а також дозволяє розкриття неструктурованих дієвих ідей, які обмежені тільки людською фантазією та творчістю. Аналітик не повинен застосовувати різні витончені методи, щоб мати можливість інтерпретувати візуалізацію даних. Візуалізація інформації — це також схема гіпотез, які можуть бути, і, як правило, є попередниками більш аналітичного або формального аналізу такого як статистичні гіпотези [15].

1.2. Експерти й огранізації які займалися візуалізацією

З часів зародження галузі візуалізації інформації і до тепер багато дослідників працюють над даною темою, шукають нові методи та удосконалюють результати попередників.

Одним із них є відомий американський вчений — Стюарт К. Кард. Він був старшим науковим співробітником Xerox PARC і одним з основоположників застосування людського чинника у взаємодії людини з комп'ютером. У 1983 році книга «Психологія взаємодії людини і комп'ютера», яку він написав у співавторстві Томасом Мораном і Алленом Ньюеллом, стала дуже впливовою в цій галузі. його дослідження присвячені галузі розвитку і підтримки науки взаємодії людини та інформації і візуально-семантичним прототипам з метою більшого їх розуміння.

Фернанда Вієгас і Мартін Ваттенберг відомі новаторською роботою в художній та соціальної візуалізації даних. Вони представляють дослідницьку групу, яка займається візуалізацію даних Google, заснували області соціального аналізу даних і були творцями «Many Eyes». У своїй новій роботі за допомогою новий можливостей браузерів Chrome, Firefox та програ-

мування у JavaScript, вони розробили інтерактивну, анімовану, дуже гарну карту США, на якій показані усі вітрові потоки країни. Їх робота була показана в музеях по всьому світу, і допомогла перетворити візуалізацію на художню практику [20].

Рис. 1.1. Карта вітрових потоків США.

На рис. 1.1 зображено приклад інтерактивної візуалізації даних. Створена карта зображає вітрові потоки США в певний момент часу, їх інтенсивність та напрям [20].

Іншими дослідниками візуалізації даних також є Джордж Фурнас — професор і заступник декана з навчальної стратегії у Школі інформації в Університеті Мічигану, Джеймс Д. Холан керує Лабораторією «Розподіленого пізнання і взаємодії людини та комп'ютера» в Університеті Каліфорнії в Сан-Дієго, Джордж Робертсон, П'єр Розенштіль, Бен Шнейдерман, Джон Томас Сташко та інші. Також темою візуального представлення інформації займаються такі організації: Міжнародний Симпозіум з графічного малю-

вання,Інструменти та техніки візуалізації інформації університету Пердью (PIVOT Lab), Університет штату Меріленд Лабораторії взаємодії людини і комп'ютера [15].

1.3. Досягнення в області візуалізації даних

Візуалізація інформації стає все більш важливою субдісципліною в HCl, фокусується на графічних механізмах, покликаних показати структуру інформації і поліпшити вартість доступу до сховищ великих даних. У друкованому вигляді, візуалізація інформації включає відображення числових даних (діаграми, графіки, кругові діаграми), комбінаторні співвідношення (креслення графіків) і географічні дані (закодовані карти). Комп'ютерні системи, такі як інформаційні візуалізатори і динамічні запити додали інтерактивність і нові методи візуалізації (3D, анімація).

Візуалізація інформації є комплексним дослідженням площі. Вона спирається на теорію інформаційного дизайну, комп'ютерної графіки, взаємодії людини з комп'ютером і когнітивної науки.

Практичне застосування візуалізації інформації в комп'ютерних програмах полягає у виборі, перетворення і представлення абстрактних даних у формі, яка полегшує взаємодію людини з метою геологічного вивчення та розуміння.

Важливими аспектами візуалізації інформації є інтерактивність і динаміка візуального уявлення. Сильні методи дозволяють користувачеві змінювати візуалізацію в реальному часі, тим самим надаючи безпрецедентну можливість сприйняття моделей і структурних відносин в абстрактних даних. Велика частина роботи в цій галузі спрямована на створення інноваційних графічних дисплеїв для складних наборів даних, таких як результати перепису, наукових даних і баз даних. Прикладом проблеми яка буде вирішена може бути, відображення сторінок на веб-сайті або файли

на жорсткому диску.

Візуалізація використовує інтерактивні візуальні уявлення даних для посилення пізнання. Це означає, що дані перетворюються в зображення, воно відображається на екрані чи в просторі. Зображення може бути змінено користувачем якщо вони продовжують працювати з ним. Ця взаємодія є важливою, оскільки дозволяє постійне перевизначення цілей, при потребі коли новий погляд в даних був накопичений. Візуалізація використовує те, що називається зовнішнім пізнання. Зовнішні ресурси використовуються для представлення даних. Люди звільняються від необхідності представляти все самостійно. Замість цього вони можуть просто подивитися на зображення [8].

1.4. Види візуалізації

Візуалізація є технікою для створення зображень, діаграм або анімації. Візуалізація через візуальні образи є ефективним способом представлення як абстрактних, так і конкретних ідей. Приклади з історії включають наскальні малюнки, єгипетські ієрогліфи, грецький геометрію і революційні методи Леонардо да Вінчі технічного малювання для інженерних і наукових цілей.

Візуалізація сьогодні постійно збільшує застосування в науці, освіті, техніці, інтерактивного мультимедіа, медицини тощо. Типовим для застосування візуалізації є сфера комп'ютерної графіки. Винахід комп'ютерної графіки може бути, найважливішою подією в візуалізації з моменту винаходу центральної перспективи в епоху Відродження. Розвиток анімації також сприяв просуванню візуалізації.

Як предмет з інформатики, наукова візуалізація є використанням інтерактивних сенсорних уявлень, як правило, візуальних, абстрактних даних для зміцнення пізнання, гіпотез, і міркування. Візуалізація даних є пов'я-

заною підкатегорією візуалізації статистичних графіків і географічних або просторових даних (як у тематичній картографії), що абстрагується в схематичній формі.

Наукова візуалізація є перетворенням, вибором або поданням даних моделювання або експериментів, з явною або неявної геометричною структурою, для аналізу і розуміння даних. Наукова візуалізація фокусується і підкреслює подання даних більш високого порядку з використанням в першу чергу графіки і техніки анімації. Це дуже важлива частина візуалізації. Традиційні сфери наукової візуалізації є візуалізація потоку, медична візуалізація, астрофізична візуалізації та хімічна візуалізації. Є кілька різних способів для наочного представлення наукових даних, з реконструкцією ізоповерхні.

Навчальна візуалізація за допомогою симулятора зазвичай, створює на комп'ютері образ тому він може бути вивченим. Це дуже корисно, коли вчення є про тему, яку важко інакше бачити, наприклад, атомна структура, тому що атоми занадто малі, щоб бути легко вивченими, недорогим і нескладним у використанні науковим обладнанням.

Системна візуалізація є новою областю візуалізації, яка об'єднує і включає в категорію існуючі методології візуалізації і додає до них розповідь, візуальні метафори (від галузі реклами) і візуальний дизайн. Вона також визнає важливість комплексної теорії систем, взаємовпливу систем і необхідність подання знань через онтологію. Системна візуалізація забезпечує глядачеві візуалізації систем здатність швидко зрозуміти складність системи. На відміну від інших візуалізацій, системна візуалізація прагне надати новий спосіб візуалізації складних систем через інтегративний підхід [3].

1.5. Інтерактивна візуалізація

Інтерактивна візуалізація є частиною графічної візуалізації в області комп'ютерної науки, яка включає вивчення того, як люди взаємодіють з комп'ютерами для створення графічних ілюстрацій інформації і як цей процес можна зробити більш ефективним. Щоб візуалізацію можна було розглядати як інтерактивну, вона повинна відповідати наступним критеріям:

— контроль деяких аспектів візуального представлення інформації, повинен бути виконуваний людиною;

— зміни внесені людиною, повинні бути відображеними своєчасно.

Один з типів інтерактивної візуалізації є віртуальна реальність (VR), де візуальне представлення інформації відбувається за допомогою введення в пристрої відображення, такі як стерео проектор.VR також характеризується використанням просторового відображення, де деякі з аспектів інформації, представленої в трьох вимірах, так що люди можуть досліджувати інформацію, неначебто були присутні (хоча це відбувається дистанційного), відповідність розмірів.

Іншим типом інтерактивної візуалізації є сумісна візуалізація, в якій кілька людей взаємодіють з тією ж комп'ютерною візуалізацією: обговорюють свої ідеї один з одним або досліджують інформацію спільно. Часто, сумісна візуалізація використовується, коли люди не можуть працювати разом [1].

1.6. Шаблони візуалізації інформації

Проводячи аналіз різних джерел, пов'язаних із темою дослідження, було сформульовано основні твердження чому візуалізація є важливою:

— Візуалізація дозволяє побачити те, що залишилось непоміченими. Будь-які дані містять інформацію, але якщо немає їх візуального

представлення можна не зрозуміти тенденцій, моделей поведінки і залежностей які утворюються.

— Візуалізація дає відповіді швидше. Дивлячись на графік, можна визначити тренд миттєво.

— Кольорове зображення досліджувати набагато легше і природніше ніж довгі рядки цифр. Суб'єктивно, ми даємо більшу перевагу інформації, яка представлена в гарній і зручній для сприйняття формі [18].

1.6.1. Загальні принципи.

Жак Бертін є всесвітньо відомим дослідником галузі візуалізації інформації. Його фундаментальна првця [6] написана понад 40 років тому, все ще залишається найбільш досконалим дослідженням цього питання.

Бертін ввів поняття зображення. В простому розумінні: якщо ми в змозі сказати відразу, що ми бачимо на картинці — то це зображення.

Є три рівні сприйняття зображень: початковий, проміжний і цілісний. Усі вони, відповідно, мають давати відповіді на такі питання:

— Що відображається в певний (конкретний) момент часу?

— Що відображається протягом певного відрізка часу?

— Яка загальна ідея того що зображається?

Добре виконана візуалізація дає миттєві відповіді на ці запитання. Погана — дозволяє їх знайти значно пізніше чи зовсім ні.

Ось деякі вагомі стандарти якості для доброї візуалізації. Користувачі повинні бути в змозі:

— вибирати і фільтрувати дані;

— надавати потрібного вигляду (список, матриця, діаграма, тощо);

— бачити зв'язки між даними (якщо вони існують);

— масштабувати і деталізувати відображені дані, в разі потреби [6].

Як бачимо, візуалізація повинна бути інтерактивною, тобто, готовою до

взаємодії з користувачем.

1.6.2. Патерни візуалізації.

Є такі базові патерни, які застосовуються для візуалізації інформації:

1.) Карти. Географічні карти, інфографічні карти, картограми, тощо. Ми бачимо їх майже кожного дня, коли шукаємо в Інтернеті інформацію, чи дивимось новини по телевізору.

2.) Лінійні часові графіки. Вони будуть використовуватися для візуалізації, що пов'язана з часом, наприклад, графік виконання проекту (діаграми Ґанта).

3.) Багато змінних. Це досить широкий клас моделей, використовується для візуалізації даних з багатьма змінними наприклад.

4.) Мережі. Використовуються для візуалізації залежностей, зв'язків та ієрархій [6].

Чудовим прикладом використання лінійних часових графіків може бути рис. 1.2 [6]. Це графік маршрутів потягів Caltrain. Час відправлення показані на осі x, станції — на осі Y. Дана візуалізація відповідає усім вищезгаданим стандартам доброї візуалізації.

Ми бачимо відразу, що станції розділені пропорційно щодо відстані між ними. Нахил лінії відображає реальну швидкість поїзда: чим крутіше лінія, тим швидший поїзд. Деякі обмежені послуги поїздів є пофарбованими в чорний колір.

Нажаль, цей графік цілком обмежений в плані взаємодії з користувачами. Ви можете тільки фільтрувати поїзди щодо їх швидкості, напрямку і днів тижня. При наведенні курсору на потяг відображається точний час прибуття. Недоліком є неспроможність збільшення інтервалів часу або відстані, виділення поїздів за типом, тощо.

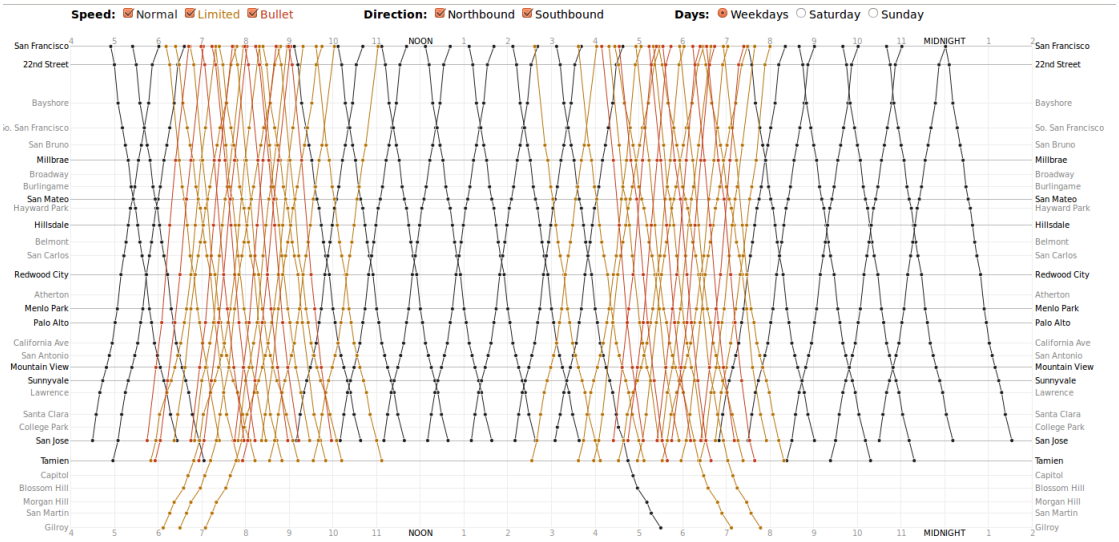

Рис. 1.2. Приклад візуалізації інформації з використанням лінійних часових графіків

1.7. Висновки до розділу 1

Здійснюючи аналітичний аналіз літературних та інших джерел було розглянуто досягнення науковців у галузі візуалізації даних. Розглянуто загальне поняття візуалізації даних, сфери застосування та перспективи розвитку.

На сьогодні візуалізація даних набуває щораз більшої популярності. Тому процес візуалізації має практичну цінність та потребує дослідження. Після проведеного дослідження можна зробити висновок, що на сьогодні не існує загального принципу(патерну) для візуалізації даних. Тому це питання є недослідженим і потребує подальшої праці.

РОЗДІЛ 2
СИСТЕМНИЙ АНАЛІЗ ТА ОБҐРУНТУВАННЯ ПРОБЛЕМИ

2.1. Системний аналіз об'єкту дослідження та предметної області

2.1.1. Побудова дерева цілей.

Для створення інформаційної системи потрібно розв'язати, виявити та впорядкувати ряд проблем. Для цього необхідно побудувати дерево цілей для цієї системи.

Дерево цілей — це графічне зображення взаємозв'язку і підпорядкованості цілей, що відображає розподіл місії і мети на цілі, підцілі, завдання та окремі дії. Метод дерева цілей є головним універсальним методом системного аналізу. Основна ідея щодо побудови дерева цілей — декомпозиція.

Декомпозиція — це метод розкриття структури системи, при якому за однією ознакою її поділяють на окремі складові.

Основне правило побудови дерева цілей — це повнота редукції — процес зведення складного явища, процесу або системи до більш простих складових. Для реалізації цього правила використовують такий системний підхід:

— ціль вищого рівня є орієнтиром, основою для розробки (декомпозиції) цілей нижчого рівня;

— цілі нижчого рівня є способами досягнення мети вищого рівня і мають бути представлені так, щоб їхня сукупність зумовлювала досягнення початкової цілі.

На рис. 2.1 зображено дерево цілей системи.

Головною метою системи є створення інформаційної системи інтерактивної візуалізації даних. Але для досягнення мети потрібно реалізувати на-

Рис. 2.1. Дерево цілей системи.

ступні цілі:

1.) Отримати вхідіні дані.

1.1.) Отримати дані про катру яка буде використовуватися для візуалізації, тобто Google Maps API.

1.2.) Отримати запит від користувача на надання інформації про радари.

2.) Стовири необхідня для візуалізації елементи.

2.1.) Розробити карту, яка буде відображатись.

2.2.) Розробити необхідні елементи для відображення, такі як: радари, елементи меню, тощо.

3.) Організувати відображення інформації

3.1.) Організувати відображення інформації для користувача так, щоб вся інформація сприймалася легко.

3.2.) Організувати можливість користувачем, при виникненні такої

необхідності, одержання детальнішої інформації.

2.1.2. Системний аналіз об'єкту дослідження та предметної області.

Системний аналіз об'єкту дослідження та предметної області, будемо здійснювати з допомогою структурного підходу. Суть структурного підходу до розроблення інформаційної системи (ІС) полягає у її декомпозиції (розбитті) на функції, що автоматизуються: система розбивається на функціональні підсистеми, які, в свою чергою поділяються на підфункції, ті – на завдання і так далі. Процес розбиття продовжується аж до конкретних процедур. При цьому система зберігає цілісний вигляд, усі її складові компоненти взаємопов'язані. Найбільшим відомим інструментальним засобом структурного аналізу є діаграми потоків даних.

Діаграми потоків даних (DFD – Data Flow Diagram) є основним засобом моделювання функціональних вимог проектованої системи. З їх допомогою ці вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, пов'язаної потоками даних. Головна мета таких засобів — продемонструвати, як кожен процес перетворить свої вхідні дані у вихідні, а також виявити відносини між цими процесами. Для представлення потоків даних, я використаю діаграму у нотації Йордана [3].

Зовнішньою сутністю в діаграмі є Користувач, який використовує дану інформаційну систему.

На рис. 2.2 зображено контекстну діаграму.

Для процесу, зображеного в діаграмі, можна здійснити декомпозицію, а саме — розбити на структурні складові, відношення між якими в тій же нотації можуть бути показані на деталізованій діаграмі рис. 2.3.

Декомпозицію процесів недоцільно здійснювати на загальній діаграмі, щоб уникнути значного збільшення кількості елементів на ній, яке приведе до ускладнення її аналізу. Тому проведемо декомпозицію на окремих

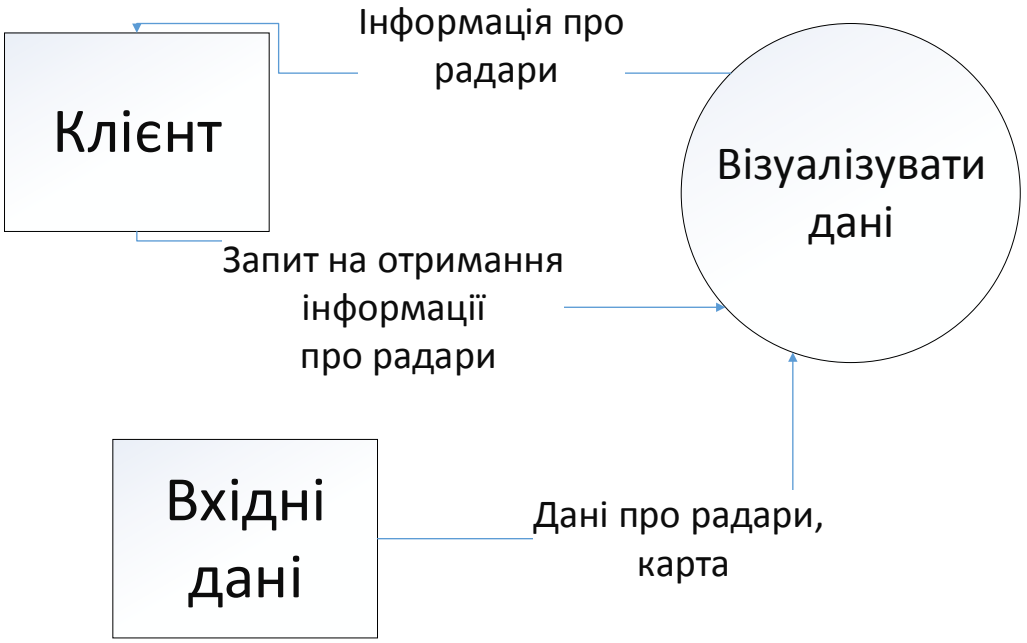

Рис. 2.2. Початкова контекстна діаграма.

Рис. 2.3. Деталізована діаграма потоків даних.

діаграмах.

На рис. 2.4 зображено деталізацію процесу "Отримати запит на візуалізацію". Як видно із діаграми, вихідною інформацією є опрацьований запит користувача про візуалізацію. Процесами, які реалізують отриманий результат є "Отримати початковий запит"і "Отримати заявку на детальну інформацію". Другий процес буде задіяний лише у випадку якщо користувач захоче побачити більш детальнішу інформацію про радари.

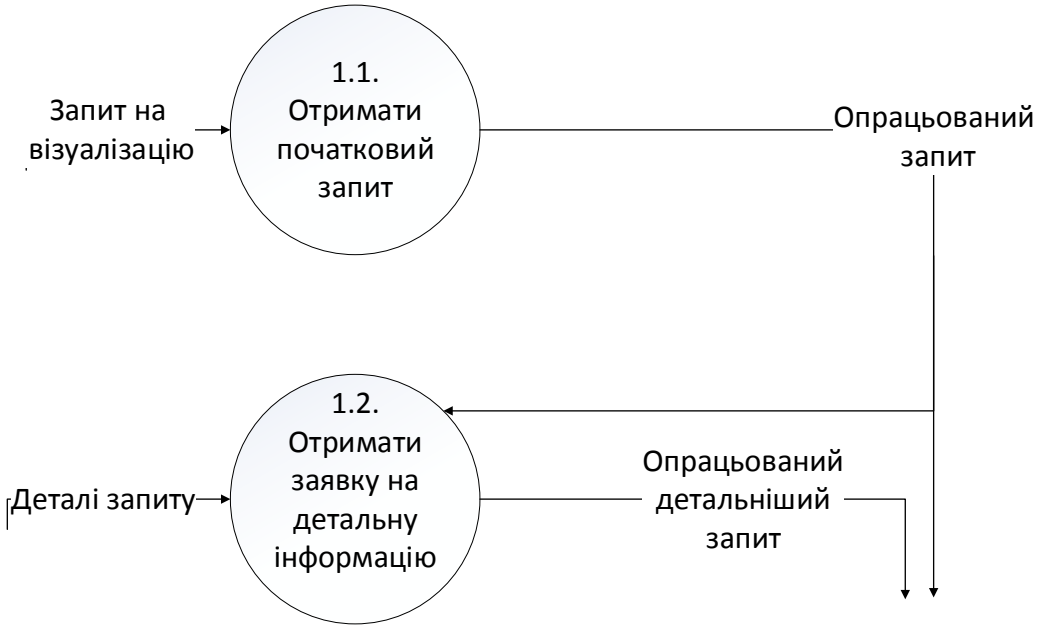

Рис. 2.4. Деталізація процесу "Отримати запит на візуалізацію".

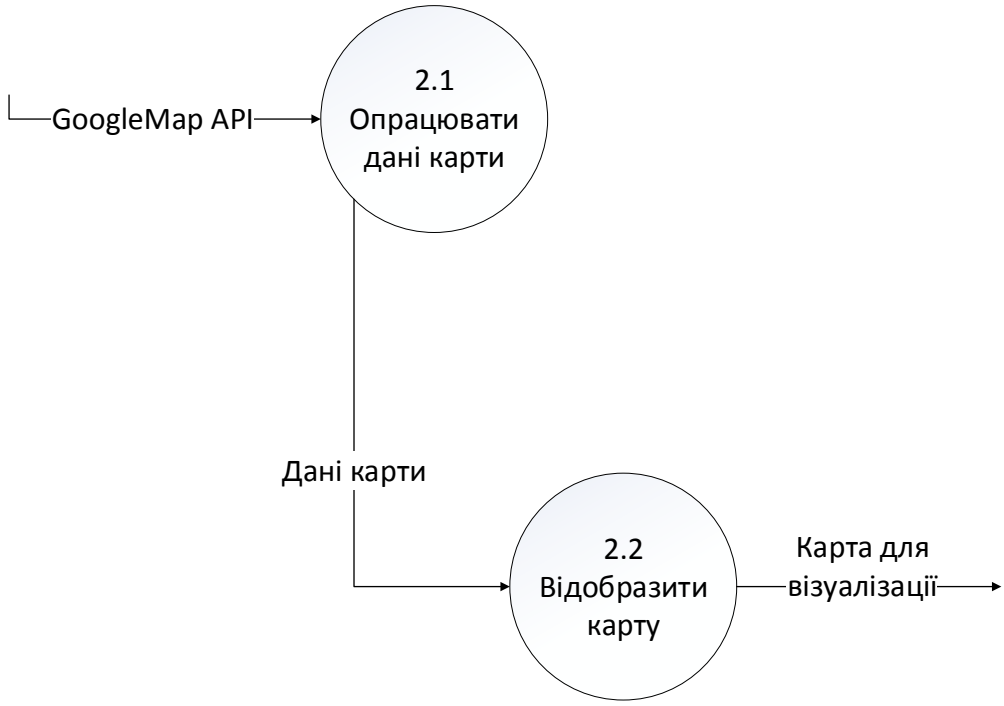

Рис. 2.5. Деталізація процесу "Отримати дані карти".

Рис. 2.5 показує деталізацію процесу "Отримати дані карти". Як видно із діаграми, вхідною інформацією є API для GoogleMaps і на виході одержуємо готову для застосування і вузуального відображення карту. Процесами, які реалізують отриманий результат є "Опрацювати дані карти"і "Відобразити карту".

На рис. 2.6 зображено деталізацію процесу "Отримати дані радарів". Як видно із діаграми, ми отримуємо вихідну інформацію з файлу, опрацьовуємо та аналізуємо її. Процесами, які реалізують отриманий результат є "Отримати дані про радари"і "Проаналізувати отримані дані". Другий процес буде задіяний лише у випадку якщо існує хоча б один радар.

На рис. 2.7 зображено деталізацію процесу "Надати візуалізацію". Як видно із діаграми, вихідною інформацією є готова для користувача візуалізація. Процесами, які реалізують отриманий результат є "Виконати початкову візуалізацію карти "Виконати початкову візуалізацію"і "Виконати

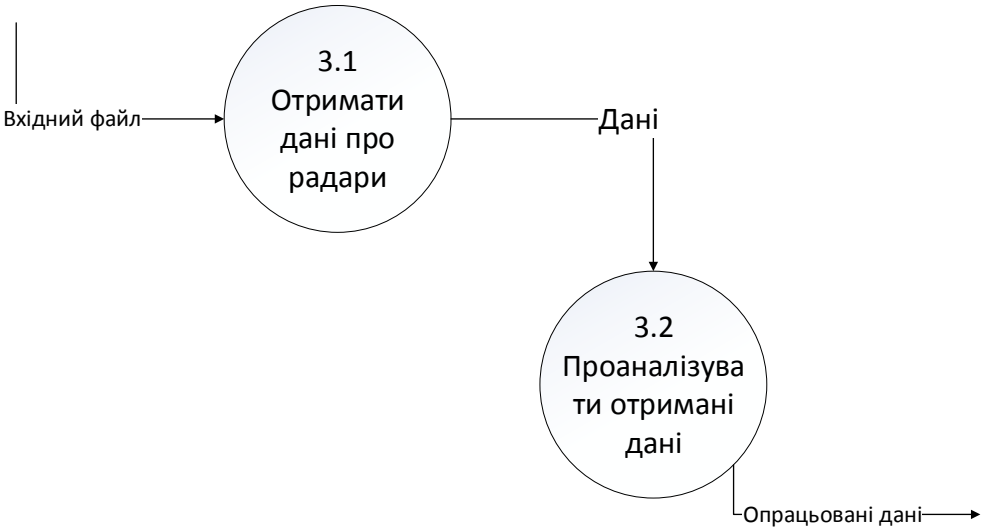

Рис. 2.6. Деталізація процесу "Отримати дані радарів".

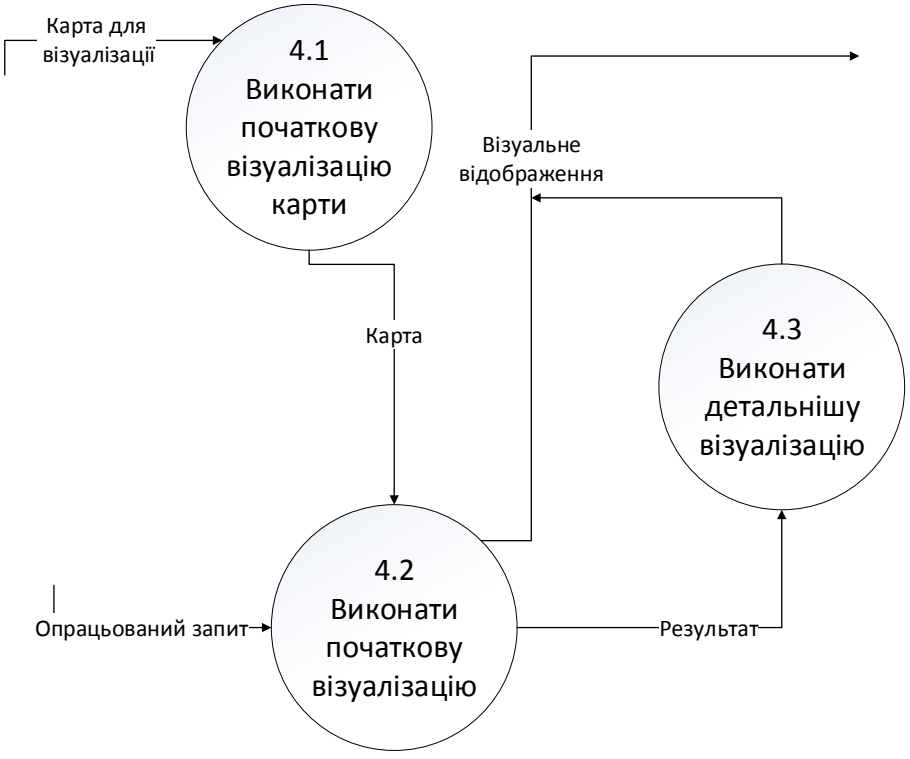

Рис. 2.7. Деталізація процесу "Надати візуалізацію візуалізацію".

детальнішу візуалізацію". Останній процес буде задіяний лише у випадку якщо користувач захоче побачити більш детальнішу інформацію про радари.

2.2. Постановка та обґрунтування проблеми

Завданням дипломної роботи є розробка інформаційної системи інтерактивної візуалізації даних та моделей.

При виконанні дипломної роботи, я поставила перед собою такі завдання:

1.) Розробити структуру для збереження вхідних даних про радар: його назва, розміщення, технічний стан, радіус дії та, у випадку несправності, прогрес виконання ремонтних робіт.
2.) Розробити алгоритм зображення карти, необхідної для подальшої візуалізації.
3.) Розробити алгоритм опрацювання інформації про радар.
4.) Забехпечити користувача наступними функціями:
    — можливість перегляду розміщення радарів на карті;
    — можливість бачити інформацію про конкретний радар у вигляді випливаючого інфовікна;
    — можливість бачити області дії та його технічний стан;
    — можливість бачити територію яка є покритою дією радарів і ту, яка є "невидимою";
    — можливість бачити статус ремонту несправного радару.
5.) Забезпечити користувача легким для сприйняття і користування функціонуванням системи.

Дана інтелектуальна інформаційна система призначення для забезпечення користувача бачити на карті інформацію про довільну кількість радарів та їх стан.

ІС розроблена під довільну операційну систему. В програмі реалізовано зберігання інформації за допомогою файлу JSON. Для забезпечення роботи даної системи необхідна наявність довільного веб-браузера. Дана система реалізовується з метою забезпечити людей візуальним відображенням інформації про радари, оскільки такий вид інформації сприймається найкраще і аналіз його є легким і природнім. Вона реалізовується з метою подальшої розробки для можливості знаходження найкращого вирішення проблеми розміщення радарів таким чином, щоб уся територія була в області дії радарів.

2.3. Висновки до розділу 2

В результаті проведеного системного аналізу інтелектуальної інформаційної системи було наведено мету функціонування системи, поставленні основні задачі, сформовано концептуальну модель, побудовано діаграму потоків даних, поставлено та обґрунтовано основні проблеми, також було реалізовано дерево цілей для графічного представлення взаємозв'язку і підпорядкованості цілей.

РОЗДІЛ 3
МЕТОДИ ТА ЗАСОБИ ВИРІШЕННЯ ПРОБЛЕМИ

3.1. Вибір та обґрунтування методів вирішення проблеми

До способів візуального або графічного представлення даних відносять графіки, діаграми, таблиці, звіти, списки, структурні схеми, карти, тощо.

Візуалізація традиційно розглядалася як допоміжний засіб при аналізі даних, проте зараз все більше досліджень говорить про її самостійну ролі.

Традиційні методи візуалізації можуть знаходити наступне застосування:

— представляти користувачеві інформацію в наочному вигляді;

— компактно описувати закономірності, притаманні початкового набору даних;

— знижувати розмірність або стискати інформацію;

— відновлювати прогалини в наборі даних;

— знаходити шуми і викиди в наборі даних.

Діаграми і графіки розсіювання часто використовуються для оцінки якості роботи того чи іншого методу.

Всі способи візуального представлення і відображення даних можуть виконувати одну з функцій:

— є ілюстрацією побудови моделі (наприклад, представлення структури (графа) нейронної мережі);

— допомагають інтерпретувати отриманий результат;

— є засобом оцінки якості побудованої моделі;

— поєднують перераховані вище функції.

Методи візуалізації, залежно від кількості використовуваних вимірю-

вань, прийнято класифікувати на дві групи:

— представлення даних в одному, двох або трьох вимірах;

— представення даних в чотирьох і більше вимірах.

До першої групи методів належать добре відомі способи відображення інформації, які доступні для сприйняття людською уявою. Практично будь-який сучасний інструмент візуалізації включає способи візуального представлення з цієї групи.

Відповідно до кількості вимірів представлення це можуть бути такі способи:

— одномірне (univariate) вимір, або 1-D;

— двовимірне (bivariate) вимір, або 2-D;

— тривимірне або проекційне (projection) вимір, або 3-D.

Слід зауважити, що найбільш природно людське око сприймає двомірні представлення інформації.

При використанні дво- і тривимірного представлення інформації користувач має можливість побачити закономірності набору даних:

— його кластерну структуру і розподіл об'єктів на класи (наприклад, на діаграмі розсіювання);

— топологічні особливості;

— наявність трендів;

— інформацію про взаємне розташування даних;

— існування інших залежностей, властивих досліджуваному набору даних.

Якщо набір даних має більше трьох вимірів, то можливі такі варіанти:

— використання багатовимірних методів представлення інформації;

— зниження розмірності до одно-, дво- або тривимірного представлення. Існують різні способи зниження розмірності, один з них — факторний аналіз. Для зниження розмірності і одночасного візуального представлення інформації на двовимірної мапі використовуються

самоорганізовані карти Кохонена.

Подання інформації в чотиривимірному і більше вимірах недоступні для людського сприйняття. Однак розроблені спеціальні методи для можливості відображення і сприйняття людиною такої інформації.

Найбільш відомі способи багатовимірного представлення інформації:

— паралельні координати;

— "обличчя Чернова";

— пелюсткові діаграми.

Перед використанням методів візуалізації необхідно:

— Проаналізувати, чи слід зображати всі дані або ж якусь їх частину.

— Вибрати розміри, пропорції та масштаб зображення.

— Вибрати метод, який може найбільш яскраво відобразити закономірності, притаманні набору даних.

Багато сучасних засобів аналізу даних дозволяють будувати сотні типів різних графіків і діаграм. Тому вибір методу візуалізації, якщо він самостійно здійснюється користувачем, не такий простий, як може здатися на перший погляд. Наявність великої кількості засобів візуалізації, представлених в інструменті, який застосовує користувач, може навіть викликати розгубленість.

Одну й ту ж інформацію можна представити за допомогою різних засобів. Для того щоб засіб візуалізації могло виконувати своє основне призначення — подавати інформацію в простому і доступному для людського сприйняття вигляді — необхідно дотримуватися законів відповідності обраного рішення змістом інформації та її функціональним призначенням. Потрібно зробити так, щоб при погляді на візуальне представлення інформації можна було відразу виявити закономірності у вихідних даних і приймати на їх основі рішення.

Серед двовимірних і тривимірних засобів найбільш широко використовуються відомі лінійні графіки, лінійні, стовпчикові, кругові секторні і ве-

кторні діаграми.

За допомогою лінійного графіка можна відобразити тенденцію, передати зміни якої-небудь ознаки в часі. Для порівняння декількох рядів чисел такі графіки наносяться на одні й ті ж осі координат.

Гістограму застосовують для порівняння значень протягом деякого періоду або ж співвідношення величин.

Кругові діаграми використовують, якщо необхідно відобразити співвідношення частин і цілого, тобто для аналізу складу або структури явищ. Складові частини цілого зображуються секторами кола. Сектори рекомендують розміщувати за їх величиною: вгорі — найбільший, решту — по руху годинникової стрілки в порядку зменшення їх величини. Кругові діаграми також застосовують для відображення результатів факторного аналізу, якщо дії всіх факторів є односпрямованим. При цьому кожен фактор відображається у вигляді одного з секторів круга.

Вибір того чи іншого засобу візуалізації залежить від поставленого завдання (наприклад, потрібно визначити структуру даних або ж динаміку процесу) і від характеру набору даних.

Для реалізації даної системи я використовуватиму метод двовимірного представлення даних [8].

3.2. Вибір та обґрунтування засобів вирішення проблеми

3.2.1. Порівняльна характерика мов програмування для веб-розробок.

Під час розробки програмного продукту велика увага віднесена на вибір мови. Із часу створення перших програмувальних машин людство придумало вже більш восьми з половиною тисяч мов програмування. Щороку їх число поповнюється новими. Деякими мовами вміє користуватися тільки невелике число їх власних розроблювачів, інші стають відомі мільйонам

людей.

PHP — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок. PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів які виконуються вже на стороні клієнта.

PHP — мова, код якої можна вбудовувати безпосередньо в html-код сторінок, які, у свою чергу, будуть коректно оброблені PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег (?>).

Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в C або Pascal.

Мова PHP здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з C, Perl. Код PHP дуже схожий на той, який зустрічається в типових програмах на C або Pascal. Це помітно знижує початкові зусилля при вивченні PHP. PHP — мова, що поєднує переваги Perl і C і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча PHP є досить молодою мовою, вона здобула таку популярність серед web-програмістів, що в наш час є мало не найпопулярнішою мовою для створення веб-застосунків (скриптів).

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і web. Важливою перевагою PHP є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте хоч би що робили розробники PHP, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність PHP достатня для створення цілком серйозних веб-застосунків.

Мова розмітки гіпертекстових документів HTML. Обмін інформацією в Інтернет здійснюється за допомогою протоколів прикладного рівня, що реалізують той або інший прикладний сервіс (пересилку файлів, гіпертекстової інформації, пошта і так далі). Одним з найбільш молодих і популярних сервісів Інтернет, розвиток якого і привело до сплеску популярності самої Інтернет, стала World Wide Web (WWW), заснована на протоколі HTTP (Hyper Text Transfer Protocol — протокол передачі гіпертекстовій інформації). Гіпертекстові документи, представлені в WWW, мають одну принципову відмінність від традиційних гіпертекстових документів — зв'язки, в них що використовуються, не обмежені одним документом, і більш того, не обмежені одним комп'ютером. Для підготовки гіпертекстових документів використовується мова HTML (Hyper Text Markup Language — мова розмітки гіпертекстових документів), що надає широкі можливості по форматуванню і структурній розмітці документів, організації зв'язків між різними документами, засоби включення графічної і мультимедійної інформації. HTML-документи є видимими за допомогою спеціальної програми — браузера. Найбільшого поширення в даний час набули браузеры Mozilla Firefox і Internet Explorer компанії Microsoft (MSIE). Реалізації Mozilla Firefox доступні практично для всіх сучасних програмних

і апаратних платформ, реалізації MSIE доступні для всіх Windows платформ, Macintosh і деяких комерційних Unix-систем.

HTML-документ складається з тексту, що є змістом документа, і тегів, що визначають його структуру і зовнішній вигляд при відображенні браузером. Тег є ключовим словом, поміщеним в кутові дужки. Розрізняють одинарні теги, як, наприклад, <p>, і парні, як <body> </body>, в останньому випадку дія тега розповсюджується тільки на текст між його відкриваючою і закриваючою дужкою. Текст всього документа полягає в теги <html>, сам документ розбивається на дві частини — заголовок і тіло. Заголовок описується тегами <head>, в яких можуть бути включені назва документа (за допомогою тегов <title>) і інші параметри, що використовуються браузером при відображенні документа. Тіло документа поміщене в теги <body> і містить власне інформацію, яку бачить користувач. За відсутності тегів форматування весь текст виводиться у вікно браузера суцільним потоком, переклади рядків, пропуски і табуляції розглядаються як пробільні символи, декілька пробільних символів, що йдуть підряд, замінюються на один.

З моменту свого створення, HTML і пов'язані з нею протоколи порівняно швидко отримали визнання. Однак, в перші роки існування цієї мови розмітки не було жодних чітких стандартів. Хоча її творці спочатку і задумували HTML як семантичну мову, позбавлену презентаційних можливостей, її практичне використання із різними браузерами призвело до додавання багатьох презентаційних елементів і атрибутів в HTML.

Останні стандарти пов'язані з HTML відображають зусилля з подолання хаотичного розвитку мови і створення раціональної основи для розробки як змістовних, так і виразних документів. Щоб повернути HTML її роль семантичної мови, Консорціум Всесвітньої павутини розробив мови стилізування такі як Каскадні таблиці стилів та Розширена мова таблиць стилів, аби перенести на них відповідальність за вигляд документу. У зв'яз-

ку з цим, специфікація HTML повільно почала повертатися виключно до семантичних елементів.

CSS. Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) — спеціальна мова, що використовується для відображення сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі. CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо. CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS). Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо.

CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою, у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових броузерах та ін.). Один і той самий HTML або XML документ може бути

відображений по-різному залежно від використаного CSS.

JavaScript (JS) — динамічна, об'єктно-орієнтована мова програмування. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів, тощо.

JavaScript класифікують як прототипну, скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером [19].

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої роз-

робки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Python портований і працює майже на всіх відомих платформах — від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android [5].

Ruby — це інтерпретована, повністю об'єктно-орієнтована мова програмування з чіткою динамічною типізацією. Мова відрізняється високою ефективністю розробки програм і увібрало в себе кращі риси Perl, Java, Python, Smalltalk, Eiffel, Ada і Lisp. Ruby поєднує в собі Perl-подібний синтаксис із об'єктно-орієнтованим підходом мови програмування Smalltalk. Також деякі риси запозичено із мов програмування Python, Lisp, Dylan та CLU.

Багатоплатформова реалізація інтерпретатора мови Ruby поширюється як вільне програмне забезпечення.

3.2.2. Методи візуалізації карт.

Manu Google — набір додатків, побудованих на основі безкоштовного картографічного сервісу і технологій, які надає компанія Google.

Сервіс являє собою карту та супутникові знімки всього світу (а також Місяця і Марса). З сервісом інтегрований бізнес-довідник і карта автомобільних доріг, з пошуком маршрутів, яка охоплює США, Канаду, Японію, Гонконг, Китай, Великобританію, Ірландію (тільки центри міст) і деякі райони Європи.

Існує можливість використовувати сервіс для створення своїх продуктів сторонніми компаніями. На сьогоднішній день це безкоштовна служба, але можливість додати рекламу залишена на майбутнє.

Для розробників сайтів зручно буде використати JavaScript для керування функціональністю карт, правда кількість запитів з одного сервера обмежена. Google Static Maps API дозволяє будувати статичні мапи за допомогою спеціальних url'ів. Також існують версії API під різні види мобільних пристроїв [12].

OpenStreetMap — це відкритий проект зі створення загальнодоступних мап світу силами спільноти. Проект заснований у Великобританії в липні 2004 року Стівом Костом (Steve Coast). У квітні 2006-го OSM зареєстровано як фонд. «Фонд OpenStreetMap — міжнародна некомерційна організація, створена для підтримки розвитку та розповсюдження геопросторових даних, а також надання можливості використання геопросторових даних будь-ким».

Сайт OpenStreetMap надає інтерфейс «ковзаючої мапи» на основі JavaScript-бібліотеки Leaflet (до 23 листопада 2012 р. — на OpenLayers), який наживо відображає мапу, використовуючи тайли, згенеровані Mapnik і тайли з інших джерел. Можна генерувати мапи локально, встановивши Mapnik та завантаживши дані.

Правити мапи можна безпосередньо у веб-переглядачі посередництвом редактора iD, HTML5-додатка, написаного з використанням D3.js компанією MapBox. Через веб-переглядач доступний також редактор Potlatch 2, написаний на Flash. JOSM та Merkaator — потужніші програми для персо-

нального комп'ютера, які краще пасуватимуть досвідченим користувачам.

Leaflet — це сучасна відкрита з вихідний кодом Java-бібліотека для мобільних інтерактивних карт. Вона розроблена Володимиром Агафонкіном з командою. При об'ємі всього близько 33 КБ JS, вона має всі функції, більшість розробникам коли-небудь може знадобитись для інтернет-карт.

Leaflet розроблена з метою простоти, продуктивності і зручності використання. Вона працює ефективно на всіх основних настільних і мобільних платформах, користуючись HTML5 та CSS3 на сучасних браузерах та все ще є доступною і на старих. Вона може бути розширеною величезною кількістю плагінів, має красивий, легкий у використанні і добре документований API і простий, легкий для читання вихідний код.

3.2.3. Ресурси й інструменти для візуалізації.

Графіки і таблиці — найкращий засіб для представлення даних в зручному для вивчення вигляді. Нижче представлений огляд з 15 бібліотек для створення різних форматів візуалізації з використанням програмного мови Javascript.

Бібліотека sigmajs призначена для малювання графів. Вона дозволяє публікувати графи на веб-сторінках і інтегрувати їх в додатки. Більшість параметрів налаштувань можна налаштувати на свій розсуд, включаючи інтерактивність графа.

BonsaiJS — бібліотека з відкритим кодом, призначена для створення графіки та анімації. В бібліотеці є функція створення простих форм, а також функція path, що дозволяє створювати власні форми. Також у користувачів є можливість застосовувати різні кольори, градації і фільтри.

Chart.js використовує Javascript і HTML5 Canvas для побудови гістограм і різних типів діаграм: кругових, полярних, кільцевих і ін. Одна з переваг Chaart.js — дизайнерське оформлення і можливість використання анімаційних ефектів.

Aristochart дозволяє будувати двомірні лінійні статичні графіки. Програма передбачає набір можливостей по дизайнерському оформленню, зміни міток, інших елементів графіка та його адаптації.

JS Charts — генераторів графіків і схем, що не вимагає великих знань в програмуванні, будь-яких плагінів або серверних модулів. JS Charts дозволяє створювати схеми на основі різних шаблонів, таких як стовпці, кругові діаграми і прості лінії. Бібліотека сумісна з більшістю веб-браузерів.

Бібліотека Highcharts JS дозволяє легко додавати інтерактивні і анімовані графіки на сайт або у веб-додатки. Підтримується безліч видів діаграм: лінійні, кругові, колонні розсіюють, тощо. Програма працює з усіма популярними видами браузерів, включаючи Safari і iPhone.

Flot — Javascript-бібліотека, що дозволяє швидко створювати динамічні графіки, сумісні з будь-якими браузерами та операційними системами. Для використання Flot буде потрібно бібліотека jQuery.

AwesomeChartJs — проста у використанні Java-sript бібліотека, за допомогою якої можна швидко створити такі види графіків: вертикальні і горизонтальні стовпці, лінійно-столбцовая графіки, кругові, кругові розсувні і кільцеві діаграми. Налаштування зовнішнього вигляду сильно обмежені.

FusionCharts — пакет побудови діаграм з відкритим вихідним кодом для побудови flash-графіків, призначених для візуалізації даних в додатках і презентаціях. Додаткові можливості FusionCharts відкриваються після оплати.

JSXGraph — розроблена в Байротському Університеті бібліотека, використовувана для відображення геометричних креслень в веб-браузері.

Arbor.js використовує бібліотеку jQuery для малювання графів прямо в браузері.

Micropolar — бібліотека інтегрована з D3.js і призначена для побудови графіків в полярних системах координат.

Morris — бібліотека з відкритим вихідним кодом, за допомогою якої мо-

жна будувати графіки на простий сітці горизонтальних ліній. Для роботи з Morris будуть потрібні бібліотеки jQuery і Raphal.

Dygraphs — ще одна бібліотека з відкритим кодом, призначена для малювання часових рядів. Графіками можна додавати елементи інтерактивності.

JFreeChart — безкоштовна бібліотека з відкритим кодом, що дозволяє створювати графіки для додатків. Програма підтримує широкий спектр типів діаграм і багато типів вихідних форматів [19].

3.2.4. Порівняльна характеристика SVG і Canvas.

HTML5 Canvas і SVG — веб-технології, які дозволяють використовувати високоякісну графіку в браузерах, але фундаментально вони докорінно відрізняються один від одного.

SVG — векторний графічний формат, заснований на XML. SVG-контент може бути статичним, динамічним, інтерактивним і анімованим - він дуже гнучкий. Також є можливим зміна оформлення SVG за допомогою CSS і визначення поведінки об'єктів за допомогою SVG DOM. І звичайно, так як текст всередині SVG зберігається у файлі, він залишається відносно доступним для використання. Також SVG-контент можна вставити безпосередньо в HTML з використанням елементу object.

За допомогою SVG можна отримати набагато більше, ніж просто векторну графіку і анімацію. Можна розробити високоінтерактівний веб-додаток з розширеними анімаційними подіями, фільтрами, тощо.

Специфікація HTML5 Canvas визначає універсальний JavaScript API, що дозволяє виконувати операції створення об'єктів. Для малювання на канві можна використовувати два різні підходи: 2D-підхід, 3D-підхід (WebGL).

Перший краще впроваджений і доступний у всіх сучасних веб-браузерах, в той час як другий знаходиться на ранній стадії визначення, маючи лише кілька експериментальних реалізацій.

Таблиця 3.1

Порівняння переваг SVG і Canvas.

Canvas	*SVG*
– Висока продуктивність при відображенні будь-яких 2D об'єктів.	– SVG дозволяє масштабувати зображення при різних розширеннях екрану.
– Стабільна продуктивність. Продуктивність падає тільки при збільшенні розширення зображення.	– SVG дуже добре підтримує анімацію. Елементи можуть бути анімовані з використанням описового синтаксису або за допомогою JavaScript.
– Можна зберегти отримане зображення в PNG або JPG файл.	– Можна отримати повний контроль над кожним елементом, використовуючи SVG DOM API в JavaScript.
– Найкраще підходить для створення растрової графіки, редагування зображень і операцій, що вимагають маніпулювання на рівні пікселів.	– SVG зберігається у форматі XML, що надає більше можливостей браузерам із забезпечення доступності SVG документів у порівнянні з елементом canvas. Таким чином, SVG виглядає кращим рішенням для користувача інтерфейсів веб-додатків.

Таблиця 3.2

Порівняння недоліків SVG і Canvas.

Canvas	*SVG*
– Піксельна побудова. – Не існує API для анімації. Доведеться використовувати таймери або інші події для поновлення канви. – Слабкі можливості по рендерингу тексту. – HTML 5 Canvas не підходить для створення веб-сайтів або інтерфейсів веб-додатків, користувацькі інтерфейси зазвичай повинні бути динамічними та інтерактивними, а Canvas вимагає від вас постійного перемальовування кожного елемента в інтерфейсі.	– Низька швидкість рендринга при збільшенні складності документа (малюнка), так як використовується модель DOM – SVG не підходить для таких додатків як ігри.

Кожна технологія має свою область застосування. HTML 5 Canvas слід використовувати для:

— редагування зображень;

— створення растрової графіки;

— аналізу зображень;

— створення ігрової графіки.

SVG слід використовувати для:

— створення користувацьких інтерфейсів веб-додатків, незалежних від розширення екрану;
— високоінтерактивних анімованих користувацьких інтерфейсів;
— графіків і діаграм;
— редагування векторних зображень.

3.3. Висновки до розділу 3

В результаті наведення методів та засобів вирішення проблеми, було здійснено огляд можливих та вибір тих які застосовуватимуться, а саме — мовою програмування яка використовується для реалізації є JavaScript. Для розмітки буде використано HTML5, а для редагування стилів CSS. Для створення зображень використовуватиметься формат SVG, оскільки він найкраще підходить для веб-додатків.

РОЗДІЛ 4
ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1. Описання реалізації завдання

4.1.1. Загальні відомості.

Для написання даної програми були застосовані такі засоби: HTML5, CSS3, JavaScript, Polymer, GoogleMaps API, Inskape і платформа Trident Genesis.

Програма складається із файлу з вихідним кодом — map.html, вхфдного файлу array.json, та файлів .svg із зображеннями які використовуються в програмі. Загальний об'єм пам'яті який використовується — 151,4 kB.

4.1.2. Функціональне призначення.

Призначенням даної програми є надати візуальне проедставлення інформації про стан радарів авіакомпанії. Користувач може бачити розміщені на карті радари. Кожен радар має назву й інфовікно для відтворення детальної інформації.

4.1.3. Описання логічної структури.

Патерни для візуалізації

Для того щоб виконати візуалізацію даних, необхідно щоб система яка буде виконувати дане завдання вирішувала такі проблеми:

1.) *Інтерфейс.* Потрібно створити зручний для використання користувачем інтерфейс із продуманими усіма, потрібними для використання системи, можливостями.

2.) *Взаємодія з користувачем.*Інтерактивність — дуже важлива складова будь-якої візуалізації. Вона дозволяє збільшити функціональ-

ність системи та покращити зручність роботи для користувача.

3.) *Стратегія подання і відображення даних.* Для того щоб система була потрібною, їй необхідно володіти певною функціонально-запотребованою стратегією.

Для того щоб створити систему візуалізації даних, необхідна реалізація класів, за допомогою який можна вирішити вищезгадані проблеми. Ці класи є описані нижче.

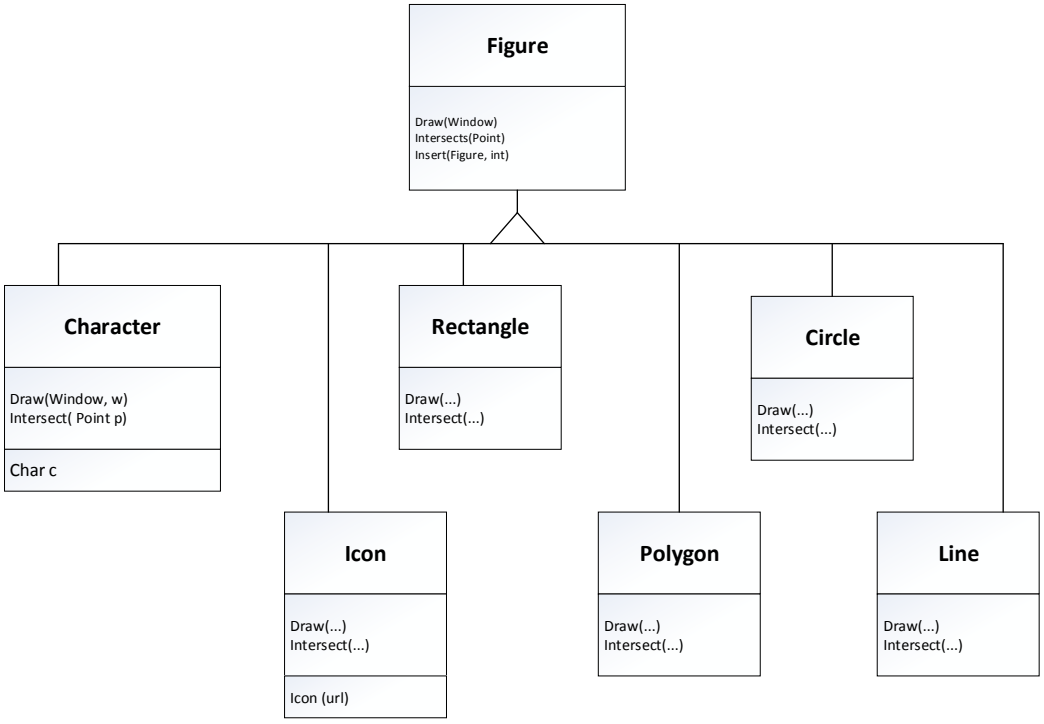

Рис. 4.1. Ієрархія класу Figure

Для того щоб можна було відображати необхідні фігури, є потрібним клас Figure. На рис. 4.1 зображена ієрархія цього класу. Він має такі основні властивості, як: намалювати себе, визначення простору який заємає, та перетин із іншими об'єктами. В цього класу є наступні субкласи: Character, Ractangle, Circle, Polygon, Line and Icon. Усі вони перевизначають функції Draw() та Intersect().

Субклас Rectangle перевизначає Draw() таким чином:

```
void Rectangle::Draw (Window* w) {
w->DrawRect(_x0, _y0, _x1, _y1);
}
```

де x0, y0, x1 і y1 є аргументами даної функції, які визначають два протилежні кути прямокутника. DrawRect — це операція, яка малює прямокутник на екрані.

Для того щоб прикрасити фігури, або щоб можна було утворювати композицію кількох фігур потрібен клас MonoFigure. На рис. 4.2 зображено взаємозв'язок класу MonoFigure.

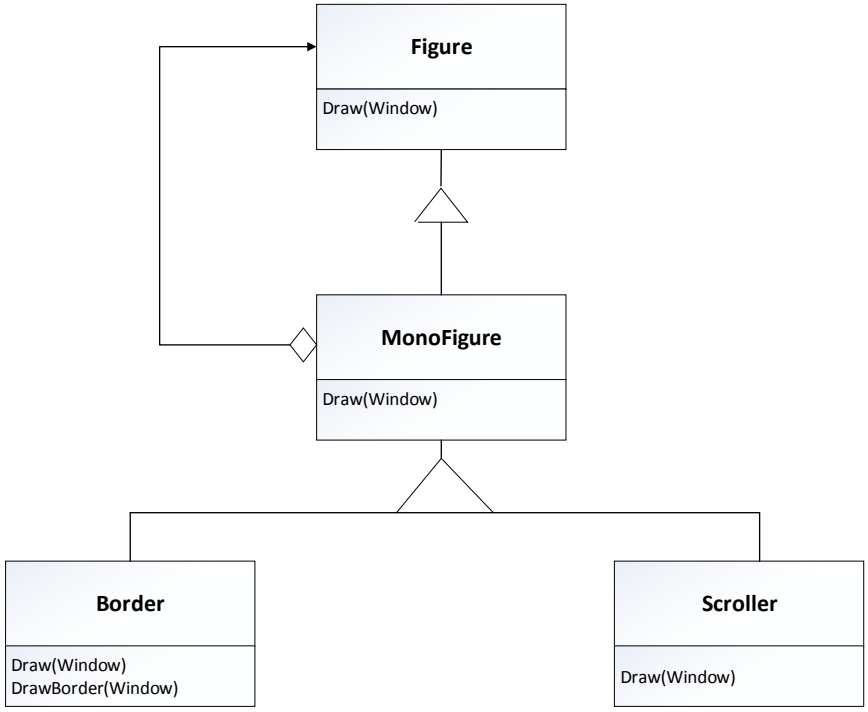

Рис. 4.2. Взаємозв'язок класу MonoFigure

Цей клас реалізує операцію Draw() таким чином:

```
void MonoFigure::Draw (Window* w) {
_component->Draw(w);
}
```

Ми визначили клас Compositor для об'єктів, які можуть інкапсулировати форматування алгоритму. Інтерфейс яким чином виконувати форма-

тування і саме. На рис. 4.3 зображено взаємозв'язок класів Compositor і Composition.

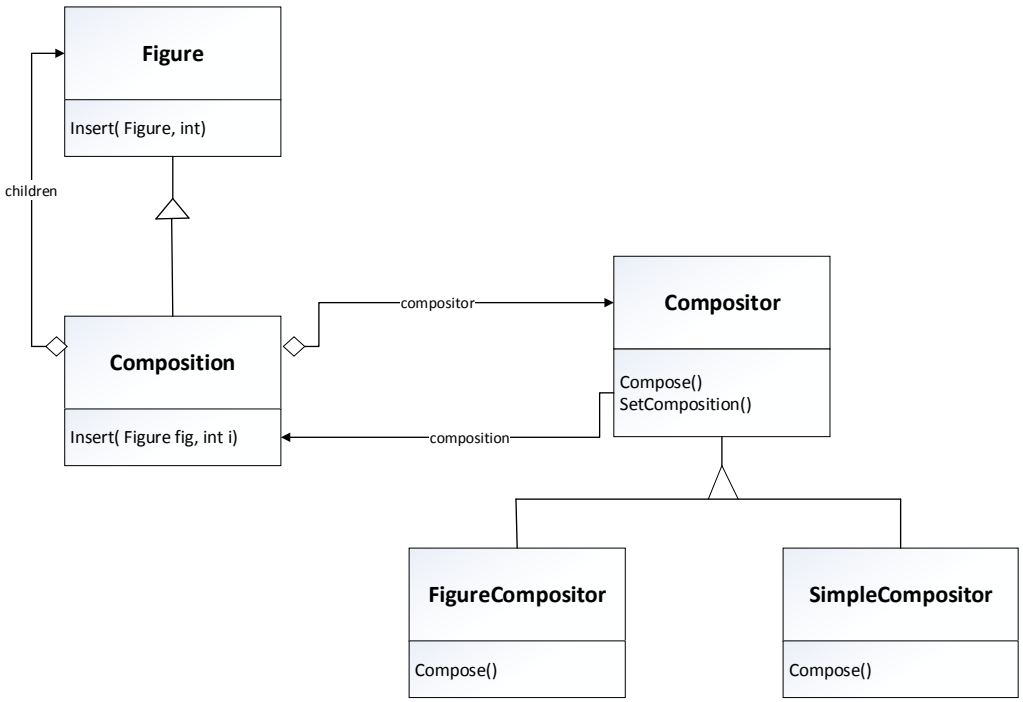

Рис. 4.3. Взаємозв'язок класів Compositor і Composition

Модель класу Decorator фіксує відносини класів і об'єктів, які підтримують елементи декору з прозорим терміном "додатки". Цей клас може ширше значення, ніж те, що ми розглянете у цій роботі. У структурі Decorator, додатки ставляться до всього, що додає відповідальності до об'єкту. Ми можемо, наприклад, прикрашати абстрактне синтаксичне дерево з семантичними діями, кінцевий автомат з новими переходами, для мережі стійких об'єктів з тегами атрибутів. Декоратор узагальнює цей підхід. На рис. 4.4 зображено взаємозв'язок класу Decorator.

Інкапсуляція об'єктів в алгоритмі є ціллю моделі класу Strategy. Ключовими учасниками в структурі є об'єктами стратегії (яка інкапсулює різні алгоритми) і контексту, в якому вони працюють.

Мтою застосування класу Strategy є розробка інтерфейсів стратегії і

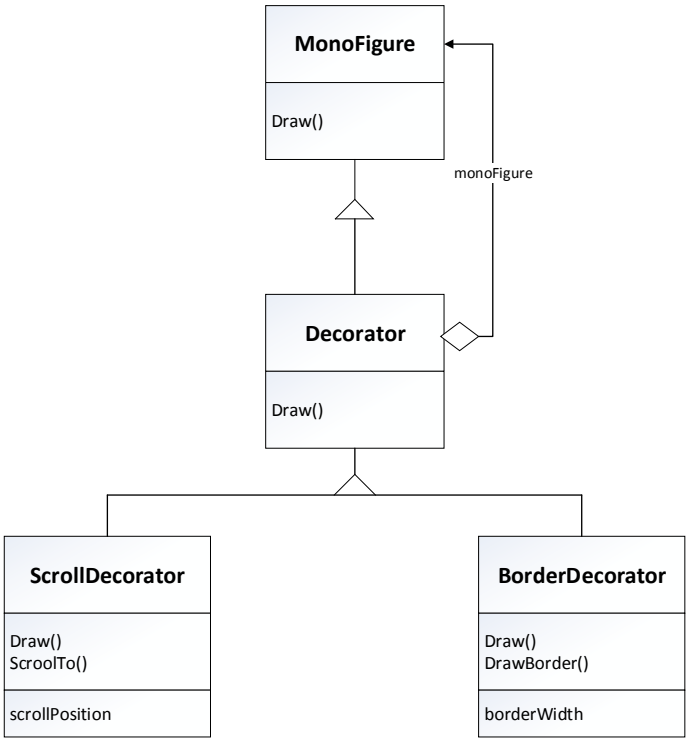

Рис. 4.4. Взаємозв'язок класу Decorator

їх контекст, що є досить загальним підтриманням алгоритмів. У даному прикладі, базова підтримка інтерфейсу Figure для доступу, вставки і видалення достатньо, змінити підклас Compositor. Вцілому фізична структура об'єкта, незалежно від алгоритму використовує список завдань для нього.

На рис. 4.5 зображено взаємозв'язок класу Strategy.

4.1.4. Запуск програми. Вхідні та вихідні дані.

4.2. Аналіз отриманих результатів

4.2.1. Аналіз результату роботи системи.

4.3. Висновки до розділу 4

При цьому було отримано такі наукові результати.

1. що, яким чином і який ефект було досягнуто;

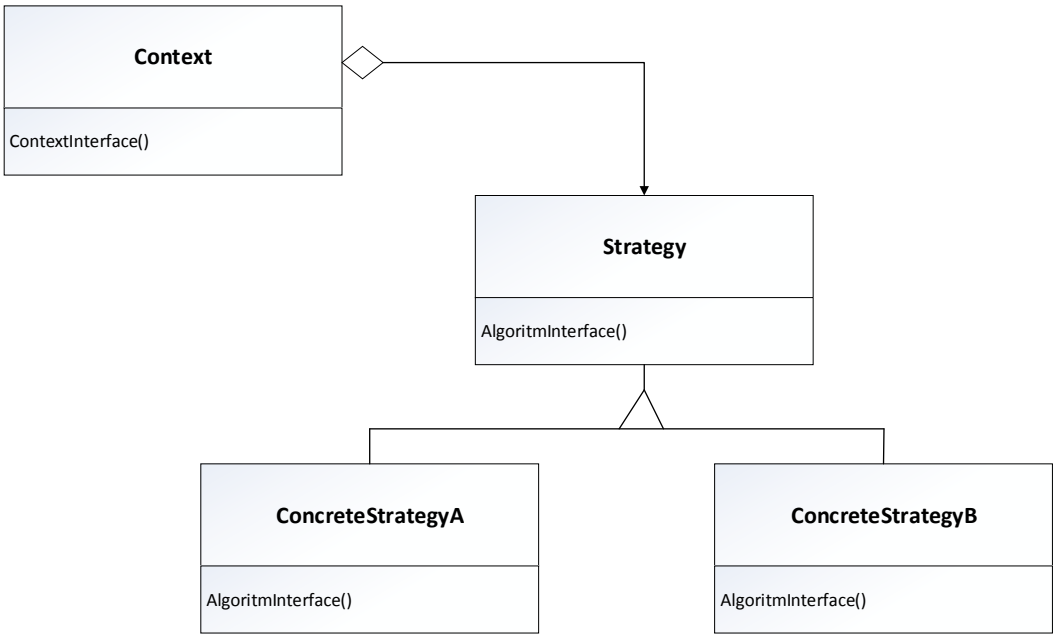

Рис. 4.5. Взаємозв'язок класу Strategy

2. що, яким чином і який ефект було досягнуто;
3. що, яким чином і який ефект було досягнуто;
4. що, яким чином і який ефект було досягнуто.

ВИСНОВКИ

У курсовій роботі вирішено актуальну науково-прикладу задачу розвитку...

При цьому було отримано такі наукові результати.

1. що, яким чином і який ефект було досягнуто;
2. що, яким чином і який ефект було досягнуто;
3. що, яким чином і який ефект було досягнуто;
4. що, яким чином і який ефект було досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *A. C.* The functional art: an introduction to information graphics and visualization / C. A. — New York, USA: New rider, 2013. — P. 180–217.
2. *A. C.* The functional art: an introduction to information graphics and visualization / C. A. — Newrider, 2013. — P. 217.
3. *B. B.* The Craft of Information Visualization: Readings and Reflections / B. B. — Maryland, 2003. — P. 340.
4. *Bauer C.* Java Persistence with Hibernate / C. Bauer, G. King. — Manning Publications, 2006. — 904 p.
5. *Beazley D.* Python Cookbook / D. Beazley, B. K. Jones. — 3rd edition edition. — O'Reilly Media, 2013. — . — P. 706.
6. *Bertin J.* Semiology of graphics, diagrams, networks, maps / J. Bertin. — First edition edition. — California: Esri, 2011. — P. 15–34.
7. *Cook D.* Interactive and Dynamic Graphics for Data Analysis / D. Cook, D. F. Swayne. — New York, USA: Springer US, 2007. — 350 p.
8. Data Visualization Tools: Electronic source. `http://visual.ly/learn/data-visualization-tools`.
9. *Dix A.* Human-Computer Interaction / A. Dix. — New York, USA: Springer US, 2009. — P. 1327–1331.
10. *Fielding R. T.* Architectural styles and the design of network-based software architectures: Phd thesis. — University of California, Irvine, USA, 2000. `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.
11. *Goetz B.* Java Concurrency in Practice / B. Goetz, T. Peierls, J. Bloch. — Addison-Wesley Professional, 2006. — 384 p.
12. Google Developers. `https://developers.google.com`.
13. Leaflet. `http://leafletjs.com`.

14. *Odersky M.* Event-based programming without inversion of control / M. Odersky // In Proc. Joint Modular Languages Conference (2006), Springer LNCS. — Springer, 2006. — P. 4–22.
15. *Owen G. S.* History of visualization / G. S. Owen // *ACM SIGGRAPH.* — 1999. — Vol. 13, no. 3. — P. 115–117.
16. Ruby. `https://www.ruby-lang.org/en`.
17. *Sears A.* Human-Computer Interaction / A. Sears, J. A. Jacko. — 2nd edition. — Tailor&Francis Group, 2009. — 352 p.
18. *Tufte E.* The Visual Display of Quantitative Information / E. Tufte. — Second edition edition. — Connecticut: Graphics Press, 2001. — P. 1–106.
19. Useful JavaScript libraries. `http://www.smashingmagazine.com/2009/03/02/40-stand-alone-javascript-libraries-for-specific-purposes`.
20. Wind map: Electronic source. `http://hint.fm/wind/`.

ABSTRACT

ДОДАТОК А

map.html

```
<!doctupe HTML>
<html>

<head>
        <title>Radars</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, minimum-scale=1.0,
          initial-scale=1.0, user-scalable=yes">

        <link rel="import" href="/resources/polymer/polymer/polymer.html">
        <link rel="import" href="/resources/polymer/font-roboto/roboto.html">
        <link rel="import" href="/resources/polymer/core-header-panel
          /core-header-panel.html">
        <link rel="import" href="/resources/polymer/core-draver-panel
          /core-draver-panel.html">
        <link rel="import" href="/resources/polymer/core-toolbar
          /core-toolbar.html">
        <link rel="import" href="/resources/polymer/core-range
          /core-range.html">
        <link rel="import" href="/resources/polymer/core-tooltip
          /core-tooltip.html">
        <link rel="import" href="/resources/polymer/core-icon-button
          /core-icon-button.html">

        <style>
                html,
                body {
                        height: 100%;
                        margin: 0;
                        background-color: #E5E5E5;
                        font-family: 'RobotoDraft', sans-serif;
                }
```

```
                google-map {
                        display: block;
                        height: 600px;
                }

                #controlsToggle {
                        position: absolute;
                        left: 10%;
                        bottom: 10%;
                }

                #map-canvas {
                        width: device-width;
                        height: 100%;
                }

                core-header-panel {
                        height: 100%;
                        overflow: auto;
                        -webkit-overflow-scrolling: touch;
                }

                core-range {
                        width: auto;
                        height: 100%;
                        overflow: auto;
                        background: green;
                        -webkit-overflow-scrolling: touch;
                }

                core-toolbar {
                        background: #03a9f4;
                        color: white;
                }
        </style>
        <script src="https://maps.googleapis.com/maps/api/js"></script>
        <script src="http://google-maps-utility-library-v3.googlecode.com/svn
/trunk/maplabel/src/maplabel.js"></script>

        <script>
```

```
                var radars = [
                        ['A', -45.67548, 168.70605, 'on', 80, 100],
                        ['B', -44.02442, 171.29883, 'broken', 50, 10],
                        ['C', -39.53794, 174.68262, 'off', 100, 100],
                        ['E', -38.51379, 177.36238, 'broken', 78, 95],
                        ['F', -41.705373, 172.35352, 'on', 80, 100],
                        ['D', -35.69299, 174.06738, 'on', 70, 100]
];

                var NZCoords = [
                                new google.maps.LatLng(-34.32622, 172.97974),

                                                ];

                var map;
                var geocoder;
                var circle;
                var redOptions;
                var blueOptions;
                var greyOptions;
                var circles = [];
                var center = new google.maps.LatLng(-41.65, 170.73);
                var bounds = new google.maps.LatLngBounds();

                function initialize() {

                        var bounds = new google.maps.LatLngBounds();
                        var infowindow = new google.maps.InfoWindow();
                        var mapCanvas = document.getElementById('map-canvas');
                        var mapOptions = {
                                center: center,
                                zoom: 6,
                                mapTypeId: google.maps.MapTypeId.ROADMAP
                        }

                        map = new google.maps.Map(mapCanvas, mapOptions)
                        var imageOff = {
                                url: 'off.svg',
                        };
                        var brokenImage = {
```

```
                url: 'broken.svg',
        };
        var imageOn = {
                url: 'on.svg',
        };


        for (i = 0; i < radars.length; i++) {
                var radar = radars[i];
                var myLatLng = new google.maps.LatLng(radar[1],
                 radar[2]);
                bounds.extend(myLatLng);
                if (radar[3] === 'broken') {
                        var marker = new google.maps.Marker({
                                position: myLatLng,
                                map: map,
                                icon: brokenImage,
                                title: radar[0],
                                info: 'Name:␣radar␣' + radar[0]
                                 + "<br>" +
                                        'Place␣(Latitude␣-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Longitude):␣' +
                                        radar[1] + ';␣' +
                                         radar[2] + "<br>" +
                                        'Range:␣' + radar[4] +
                                        "␣miles" + "<br>" +
                                         'Technical␣status:␣' +
                                          radar[3]
                        });
                } else if (radar[3] === 'off') {

                        var marker = new google.maps.Marker({
                                position: myLatLng,
                                map: map,
                                icon: imageOff,
                                title: radar[0],
                                info: 'Name:␣radar␣' + radar[0]
                                 + "<br>" +
                                        'Place␣(Latitude␣-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Longitude):␣' +
```

```
                                                    radar [1] + ';␣' +
                                                     radar [2] + "<br>" +
                                                    'Range:␣' + radar [4] +
                                                    "␣miles" + "<br>"
                                                    + 'Technical␣status:␣'
                                                     radar [3]
                });
            } else if (radar[3] === 'on') {

                var marker = new google.maps.Marker({
                        position: myLatLng,
                        map: map,
                        icon: imageOn,
                        title: radar[0],
                        info: 'Name:␣radar␣' + radar[0]
                         + "<br>" +
                                                    'Place␣(Latitude␣-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Longitude):␣' +
                                                    radar [1] + ';␣' +
                                                     radar [2] + "<br>" +
                                                    'Range:␣' + radar [4] +
                                                    "␣miles" + "<br>"
                                                     + 'Technical␣status:␣
                                                      + radar [3]
                });
            }
            if (radar[3] === 'on') {
                var position = new google.maps.LatLng(rad
                + 0.09, radar[2] - 0.05);
                addAnimatedCircles(position);
            }
            google.maps.event.addListener(marker, 'click', fun
                infowindow.setContent(this.info);
                infowindow.open(map, this);
            });

        }

    }
```

```
function showProgressBar() {
        geocoder = new google.maps.Geocoder();
        for (var i = 0; i < radars.length; i++) {
                var radar = radars[i];
                if (radar[3] === "broken") {
                        var shadow;
                        var x = radar[1] + 0.47;
                        var y = radar[2] + 0.55;
                        var shadowCoords = [
                                new google.maps.LatLng(x,
                                y),
                                new google.maps.LatLng(x,
                                y + 5),
                                new google.maps.LatLng(x +
                                0.6, y + 5),
                                new google.maps.LatLng(x +
                                0.6, y),
                                new google.maps.LatLng(x,
                                y)

                                  ];

                        shadow = new google.maps.Polygon({
                                paths: shadowCoords,
                                strokeColor: '#CFD8DC',
                                strokeOpacity: 0.8,
                                strokeWeight: 0,
                                fillColor: '#90A4AE',
                                fillOpacity: 0.5
                        });

                        shadow.setMap(map);

                        var re?t1;
                        var x1 = radar[1] + 0.5;
                        var y1 = radar[2] + 0.5;
                        var rect1Coords = [
                                new google.maps.LatLng(x1,
```

```
            y1),
            new google.maps.LatLng(x1,
            y1 + 5),
            new google.maps.LatLng(x1 +
            0.6, y1 + 5),
            new google.maps.LatLng(x1 +
             0.6, y1),
            new google.maps.LatLng(x1,
            y1)

              ];

re?t1 = new google.maps.Polygon({
            paths: rect1Coords,
            strokeColor: '#CFD8DC',
            strokeOpacity: 0.8,
            strokeWeight: 1,
            fillColor: '#FFFDE7',
            fillOpacity: 1
});

re?t1.setMap(map);

var re?t2;
var x2 = radar[1] + 0.7;
var y2 = radar[2] + 0.7;
var rect2Coords = [
            new google.maps.LatLng(x2,
            y2),
            new google.maps.LatLng(x2,
            y2 + 4.5),
            new google.maps.LatLng(x2 +
            0.2, y2 + 4.5),
            new google.maps.LatLng(x2 +
            0.2, y2),
            new google.maps.LatLng(x2,
            y2)

                              ];
```

```
re?t2 = new google.maps.Polygon({
        paths: rect2Coords,
        strokeColor: '#CFD8DC',
        strokeOpacity: 0.8,
        strokeWeight: 1,
        fillColor: '#B0BEC5',
        fillOpacity: 1
});

re?t2.setMap(map);

var re?t3;
var x3 = radar[1] + 0.7;
var y3 = radar[2] + 0.7;
var rect3Coords = [
        new google.maps.LatLng(x3,
        y3),
        new google.maps.LatLng(x3,
        y3 +
        (4.5 * (radar[5] / 100))),
        new google.maps.LatLng(x3 +
        0.2, y3 +
        (4.5 * (radar[5] / 100))),
        new google.maps.LatLng(x3 +
        0.2, y3),
        new google.maps.LatLng(x3,
        y3)

        ];

re?t3 = new google.maps.Polygon({
        paths: rect3Coords,
        strokeColor: '#80CBC4',
        strokeOpacity: 0.8,
        strokeWeight: 1,
        fillColor: '#00BCD4',
        fillOpacity: 1

});
```

```
                re?t3.setMap(map);
                var mapLabel = new MapLabel({
                        text: 'Repair status - ' +
                        radar[5] + ' %.',
                        position: new google.maps.LatLng
                        + 0.7, y3 + 1),
                        map: map,
                        fontSize: map.zoom * 3.5,
                        strokeWeight: 0,
                        fontFamily: 'RobotoDraft',
                        align: 'left',
                        fontColor: "#006064"
                });

            }
        }
    }

    function addAnimatedCircles(position) {
        var smallCircle = new google.maps.Circle({
                center: position,
                radius: 1000,
                strokeColor: "#009688",
                strokeOpacity: 1,
                strokeWeight: 1,
                fillColor: "#009688",
                fillOpacity: 0
        });
        var bigCircle = new google.maps.Circle({
                center: position,
                radius: 4000,
                strokeColor: "#40C4FF",
                strokeOpacity: 1,
                strokeWeight: 1,
                fillColor: "#40C4FF",
                fillOpacity: 0
        });
        var newCircle = new google.maps.Circle({
                center: position,
                radius: 8000,
```

```
                strokeColor: "#B3E5FC",
                strokeOpacity: 1,
                strokeWeight: 1,
                fillColor: "#B3E5FC",
                fillOpacity: 0
        });
        smallCircle.setMap(map);
        bigCircle.setMap(map);
        newCircle.setMap(map);
        var direction = 1;
        var rMax = 90000;
        setInterval(function () {
                var radius1 = bigCircle.getRadius();
                var radius2 = smallCircle.getRadius();
                var radius3 = newCircle.getRadius();

                if (radius1 > rMax) {
                        radius1 = 1000;
                }
                if (radius2 > rMax) {
                        radius2 = 4000;
                }
                if (radius3 > rMax) {
                        radius3 = 8000;
                }
                smallCircle.setRadius(radius2 + direction *
                 3000);
                bigCircle.setRadius(radius1 + direction *
                3000);
                newCircle.setRadius(radius3 + direction *
                3000);
        }, 50);
}

function drawCircle(point, radius, dir) {
        var d2r = Math.PI / 180;
        var r2d = 180 / Math.PI;
        var earthsradius = 3963;
        var points = 32;
```

```
                var rlat = (radius / earthsradius) * r2d;
                var rlng = rlat / Math.cos(point.lat() * d2r);

                var extp = new Array();
                if (dir == 1) {
                        var start = 0;
                        var end = points + 1
                } else {
                        var start = points + 1;
                        var end = 0
                }
                for (var i = start;
                        (dir == 1 ? i < end : i > end); i = i + dir) {
                        var theta = Math.PI * (i / (points / 2));
                        ey = point.lng() + (rlng * Math.cos(theta));
                        ex = point.lat() + (rlat * Math.sin(theta));
                        extp.push(new google.maps.LatLng(ex, ey));
                        bounds.extend(extp[extp.length - 1]);
                }
                return extp;
        }

        function seeNoVisibility() {

                var info = [];
                for (var i = 0; i < radars.length; i++) {
                        var radar = radars[i];
                        if (radar[3] === 'on') {
                                info.push(radar)
                        }
                }

                var options = {
                        strokeColor: '#FF0000',
                        strokeOpacity: 0.8,
                        strokeWeight: 0,
                        fillColor: '#FF0000',
                        fillOpacity: 0.35,
                        map: map,
```

```
                paths: [
                        NZCoords, drawCircle(new
                         google.maps.LatLng(info[1][1],
                        info[1][2]), info[1][4], 1),
                        drawCircle(new google.maps.LatLng
                        (info[2][1], info[2][2]),
                        info[2][4], 1),
                        drawCircle(new google.maps.LatLng
                        (info[0][1], info[0][2]),
                        info[0][4], 1),


                        ]
        }
        circle = new google.maps.Polygon(options);
        map.fitBounds(bounds);

}


function addAllCircles() {

        for (var i = 0; i < radars.length; i++) {
                var radar = radars[i];
                redOptions = {
                        strokeColor: '#FF0000',
                        fillColor: '#FF0000',
                        strokeOpacity: 0.6,
                        strokeWeight: 2,
                        fillOpacity: 0.25,
                        map: map,
                        paths: [drawCircle(new
                         google.maps.LatLng(radar[1],
                         radar[2]), radar[4], -1)]
                };

                blueOptions = {
                        strokeColor: '#2196F3',
                        fillColor: '#2196F3',
                        strokeOpacity: 0.6,
```

```
                strokeWeight: 2,
                fillOpacity: 0.25,
                map: map,
                paths: [drawCircle(new
                 google.maps.LatLng(radar[1],
                radar[2]), radar[4], -1)]
            };
            greyOptions = {
                strokeColor: '#90A4AE',
                fillColor: '#90A4AE',
                strokeOpacity: 0.6,
                strokeWeight: 2,
                fillOpacity: 0.25,
                map: map,
                paths: [drawCircle(new
                 google.maps.LatLng(radar[1],
                radar[2]), radar[4], -1)]
            };
            if (radar[3] === 'on') {
                circle = new
                 google.maps.Polygon(blueOptions);
                map.fitBounds(bounds);
            } else if (radar[3] === 'broken') {
                circle = new
                 google.maps.Polygon(redOptions);
                map.fitBounds(bounds);
            } else if (radar[3] === 'off') {
                circle = new
                 google.maps.Polygon(greyOptions);
                map.fitBounds(bounds);
            }
        }
        options1 = {
            strokeColor: '#FF0000',
            fillColor: '#FF0000',
            strokeOpacity: 0.6,
            strokeWeight: 2,
            fillOpacity: 0.35,
            map: map,
            paths: [drawCircle(new google.maps.LatLng
```

```
            (-36.52, 185.49), 15, -1)]
    };
    circle = new google.maps.Polygon(options1);
    map.fitBounds(bounds);
    options2 = {
            strokeColor: '#2196F3',
            fillColor: '#2196F3',
            strokeOpacity: 0.6,
            strokeWeight: 2,
            fillOpacity: 0.35,
            map: map,
            paths: [drawCircle(new google.maps.LatLng
            (-37.52, 185.49), 15, -1)]
    };
    circle = new google.maps.Polygon(options2);
    map.fitBounds(bounds);
    options3 = {
            strokeColor: '#90A4AE',
            fillColor: '#90A4AE',
            strokeOpacity: 0.6,
            strokeWeight: 2,
            fillOpacity: 0.35,
            map: map,
            paths: [drawCircle(new google.maps.LatLng
            (-38.52, 185.49), 15, -1)]
    };
    circle = new google.maps.Polygon(options3);
    map.fitBounds(bounds);
    var mapLabel = new MapLabel({
            text: 'Technical status of radar',
            position: new google.maps.LatLng
            (-35.7, 185.49),
            map: map,
            fontSize: map.zoom * 3.5,
            strokeWeight: 0,
            fontFamily: 'RobotoDraft',
            align: 'left',
            fontColor: "#0277BD"
    });
    var mapLabelBroken = new MapLabel({
```

```
                text: 'broken',
                position: new google.maps.LatLng
                (-36.4, 186),
                map: map,
                fontSize: map.zoom * 3.4,
                strokeWeight: 0,
                fontFamily: 'RobotoDraft',
                align: 'left',
                fontColor: "#0277BD"
        });
        var mapLabelOn = new MapLabel({
                text: 'on',
                position: new google.maps.LatLng
                (-37.4, 186),
                map: map,
                fontSize: map.zoom * 3.4,
                strokeWeight: 0,
                fontFamily: 'RobotoDraft',
                align: 'left',
                fontColor: "#0277BD"
        });
        var mapLabelOff = new MapLabel({
                text: 'off',
                position: new google.maps.LatLng
                (-38.4, 186),
                map: map,
                fontSize: map.zoom * 3.4,
                strokeWeight: 0,
                fontFamily: 'RobotoDraft',
                align: 'left',
                fontColor: "#0277BD"
        });

}

function addWorkingCircles() {

        for (var i = 0; i < radars.length; i++) {
                var radar = radars[i];
```

```
                    blueOptions = {
                            strokeColor: '#2196F3',
                            fillColor: '#2196F3',
                            strokeOpacity: 0.6,
                            strokeWeight: 2,
                            fillOpacity: 0.25,
                            map: map,
                            paths: [drawCircle(new
                             google.maps.LatLng(radar[1],
                             radar[2]), radar[4], -1)]
                    };
                    if (radar[3] === 'on') {
                            circle = new
                             google.maps.Polygon(blueOptions);
                            map.fitBounds(bounds);
                            circles.push(circle);
                    }
            }
    }
    google.maps.event.addDomListener(window, 'load', initialize);
</script>

<body fullbleed unresolved>
        <core-header-panel main rightDrawer>
                <core-toolbar>
                    <core-tooltip label="See the range of
                                    all radars">
                            <core-icon-button id="addCircles"
                            onclick=
                            "addAllCircles()" icon="cloud">
                            </core-icon-button>
                    </core-tooltip>
                    <core-tooltip label="See the range of working
                                    radars">
                            <core-icon-button id="addCircles"
                             onclick=
                            "addWorkingCircles()" icon=
                            "cloud-done">
                            </core-icon-button>
                    </core-tooltip>
```

```
                        <core-tooltip label="Area with no visibility">
                                <core-icon-button id="addCircles"
                                onclick=
                                "seeNoVisibility()" icon=
                                "visibility-off">
                                </core-icon-button>
                        </core-tooltip>
                        <core-tooltip label="The progress of repair">
                                <core-icon-button id="addCircles"
                                onclick=
                                "showProgressBar()" icon="settings">
                        </core-tooltip>
                        <core-tooltip label="Close all">
                                <core-icon-button id="addCircles"
                                 onclick=
                                "initialize()" icon="close">
                                </core-icon-button>
                        </core-tooltip>

                </core-toolbar>

                <div id="map-canvas"></div>

        </core-header-panel>

    </body>
</head>

</html>
```

ДОДАТОК В

radar.svg

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>


<svg
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
   xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
   width="50"
   height="50"
   id="svg2"
   version="1.1"
   inkscape:version="0.48.4 r9939"
   sodipodi:docname="off.svg">
  <defs
     id="defs4">
    <inkscape:perspective
       sodipodi:type="inkscape:persp3d"
       inkscape:vp_x="0 : 100 : 1"
       inkscape:vp_y="0 : 1000 : 0"
       inkscape:vp_z="180 : 100 : 1"
       inkscape:persp3d-origin="90 : 66.666667 : 1"
       id="perspective7431" />
    <clipPath
       clipPathUnits="userSpaceOnUse"
       id="clipPath3020">
      <path
         inkscape:connector-curvature="0"
         d="m 0,2400 2400,0 L 2400,0 0,0 0,2400 z"
         id="path3022" />
```

```
    </clipPath>
    <clipPath
       clipPathUnits="userSpaceOnUse"
       id="clipPath3048">
      <path
         inkscape:connector-curvature="0"
         d="m 0,2400 2400,0 L 2400,0 0,0 0,2400 z"
         id="path3050" />
    </clipPath>
    <filter
       inkscape:collect="always"
       id="filter4752"
       x="-0.76070321"
       width="2.5214064"
       y="-1.6164943"
       height="4.2329884"
       color-interpolation-filters="sRGB">
      <feGaussianBlur
         inkscape:collect="always"
         stdDeviation="80.824713"
         id="feGaussianBlur4754" />
    </filter>
  </defs>
  <sodipodi:namedview
     id="base"
     pagecolor="#ffffff"
     bordercolor="#666666"
     borderopacity="1.0"
     inkscape:pageopacity="0.0"
     inkscape:pageshadow="2"
     inkscape:zoom="7.9999996"
     inkscape:cx="23.481259"
     inkscape:cy="33.731514"
     inkscape:document-units="px"
     inkscape:current-layer="layer1"
     showgrid="false"
     inkscape:window-width="1615"
     inkscape:window-height="1026"
     inkscape:window-x="65"
     inkscape:window-y="24"
```

```
    inkscape:window-maximized="1">
  <inkscape:grid
     type="xygrid"
     id="grid7338"
     empspacing="5"
     visible="true"
     enabled="true"
     snapvisiblegridlinesonly="true" />
</sodipodi:namedview>
<metadata
    id="metadata7">
  <rdf:RDF>
    <cc:Work
       rdf:about="">
      <dc:format>image/svg+xml</dc:format>
      <dc:type
         rdf:resource="http://purl.org/dc/dcmitype/StillImage" />
      <dc:title></dc:title>
    </cc:Work>
  </rdf:RDF>
</metadata>
<g
    inkscape:label="Layer 1"
    inkscape:groupmode="layer"
    id="layer1"
    transform="translate(0,-1002.3622)">
  <g
     id="g3125-0"
     transform="matrix(1,0,0,1.125,508.39376,-39.788112)"
     style="fill:#d7eef4;filter:url(#filter4752)">
    <g
       style="fill:#afe9af"
       transform="matrix(0.32831515,0,0,-0.23291542,268.80842,988.41392)"
       id="g3144-6">
      <path
         inkscape:connector-curvature="0"
         id="path3146-9"
         style="fill:#afe9af;fill-opacity:1;fill-rule:nonzero;stroke:none"
         d="m -250.37583,-29.327317 c -258.89759,0 -426.41956,0 -654.85861,0 0,64.40
         sodipodi:nodetypes="ccccc" />
```

```
    </g>
    <g
       transform="matrix(0.30199321,0,0,-0.06625208,201.16885,1032.5216)"
       id="g3148-2"
       style="fill:#d7eef4" />
  </g>
  <g
     id="g3062"
     transform="translate(-4.7233713e-7,-1.0266357e-5)">
    <rect
       transform="matrix(0.99997474,0.00710817,-0.00390874,0.99999236,0,0)"
       style="fill:#6f8a91;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
       id="rect4325"
       width="15.017218"
       height="10.226317"
       x="16.836042"
       y="1032.1254"
       rx="0"
       ry="0" />
    <path
       style="fill:#2c89a0;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
       d="m 18.501325,1009.0502 c -6.867533,0.3105 -8.5449553,-0.1805 -11.3826162,8
       id="path4195"
       inkscape:connector-curvature="0"
       sodipodi:nodetypes="ccccccc" />
    <rect
       style="fill:#b7c4c8;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
       id="rect4199"
       width="36.72625"
       height="14.782421"
       x="484.91583"
       y="872.17004"
       rx="18.363125"
       ry="7.3912106"
       transform="matrix(0.87522272,0.48372016,-0.4704071,0.88244952,0,0)" />
    <rect
       style="fill:#006680;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
       id="rect4199-5"
       width="34.266262"
       height="11.796689"
```

```
    x="752.48859"
    y="872.98004"
    rx="17.133131"
    ry="5.898345"
    transform="matrix(0.85118216,0.52487039,-0.71427802,0.69986207,0,0)" />
<rect
    style="fill:#164450;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4199-4"
    width="3.4223363"
    height="3.2155259"
    x="-80.808762"
    y="1005.3224"
    rx="1.7111682"
    ry="1.6077629"
    transform="matrix(0.99418721,-0.10766514,0.11226849,0.99367791,0,0)" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226"
    width="1.0659527"
    height="16.713467"
    x="590.73981"
    y="906.87384"
    rx="0"
    ry="0"
    transform="matrix(0.90336227,0.42887832,-0.55183909,0.83395061,0,0)" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226-9-2"
    width="0.6422565"
    height="19.011673"
    x="54.733349"
    y="1045.1429"
    rx="0"
    ry="0"
    transform="matrix(0.75126889,-0.65999625,-0.0066552,0.99997785,0,0)" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226-9"
    width="0.46349248"
    height="2.4574933"
```

```
    x="50.534252"
    y="1010.3499"
    rx="0"
    ry="0"
    transform="matrix(0.95311735,0.30260091,-0.01391901,0.99990313,0,0)"
    inkscape:transform-center-x="-2.5972841"
    inkscape:transform-center-y="0.66060622" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226-9-4"
    width="0.60115892"
    height="5.5004401"
    x="622.349"
    y="922.90063"
    rx="0"
    ry="0"
    transform="matrix(0.903196,0.42922837,-0.57968938,0.81483754,0,0)"
    inkscape:transform-center-x="-1.8859479"
    inkscape:transform-center-y="0.46605286" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226-9-1"
    width="0.44616458"
    height="2.5947595"
    x="919.8056"
    y="401.00327"
    rx="0"
    ry="0"
    transform="matrix(0.41080269,0.91172427,-0.89601495,0.44402388,0,0)"
    inkscape:transform-center-x="-0.73165935"
    inkscape:transform-center-y="1.2069515" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226-9-4-4"
    width="0.55688888"
    height="3.6410625"
    x="450.99173"
    y="933.36597"
    rx="0"
    ry="0"
```

```
    transform="matrix(0.92363534,0.38327244,-0.41846011,0.90823518,0,0)"
    inkscape:transform-center-x="-0.85351333"
    inkscape:transform-center-y="0.16575613" />
<rect
    transform="matrix(0.99997474,0.00710817,-0.00390874,0.99999236,0,0)"
    style="fill:#2c89a0;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4325-2"
    width="18.579636"
    height="7.0632935"
    x="15.109059"
    y="1039.8784"
    rx="3.5094826"
    ry="1.752167" />
<rect
    transform="matrix(0.99997474,0.00710817,-0.00390874,0.99999236,0,0)"
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4325-2-8"
    width="20.985283"
    height="6.0038533"
    x="13.982347"
    y="1041.0139"
    rx="1.8322461"
    ry="1.4893554" />
<rect
    transform="matrix(0.99997474,0.00710817,-0.00390874,0.99999236,0,0)"
    style="fill:#53676c;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4325-2-8-3"
    width="20.985283"
    height="1.6876165"
    x="13.970608"
    y="1045.3245"
    rx="0.23397146"
    ry="0" />
<rect
    style="fill:#004455;fill-rule:evenodd;stroke:#000000;stroke-width:0;stroke-li
    id="rect4226-9-2-3"
    width="0.6422565"
    height="19.011673"
    x="1233.4874"
    y="1233.9254"
```

```
            rx="0"
            ry="0"
            transform="matrix(0.92565769,0.37836204,-0.89812864,0.43973282,0,0)" />
        </g>
    </g>
</svg>
```