МАГІСТЕРСЬКА РОБОТА

МР.ІПм – 21.00.00.000 ПЗ

Група ІПм-22-4

Арабчук Іван

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Арабчук Іван Васильович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та засоби контролю доступу до статичної та динамічної інформації

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Арабчук І.В.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Гобир Лідія Мирославівна, асистент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

_____

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**

доц. В.В. Бандура

“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Арабчуку Івану Васильовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та засоби контролю доступу до статичної та динамічної інформації”

керівник проекту (роботи) Гобир Лідія Мирославівна асистент

затверджені наказом закладу вищої освіти від “ 18 ” грудня 2023 р. № 738/7

2. Строк подання студентом проекту (роботи) 25 січня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій певного класу

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження та аналіз методів і засоби імплементації соціальної взаємодії отримання статичних та динамічних даних

2. Моделі та засоби імплементації соціальної взаємодії

3. Оцінювання отриманих імплементацій соціальної взаємодії

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Систематичний процес картографування (рис. 1.3)

2. Концептуальна модель SMADL, її версії, створені за допомогою розширення (рис. 2.10)

3. Інтерфейс uCloud Console (рис. 2.22)

4. Ієрархічна модель (рис. 3.1)

5. Фрагмент класифікації соціальних мереж (СМ) та їх сервісів (рис. 3.2)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц., к.т.н. Вовк Р.Б.	
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	01.10.2023	виконано
2	Методи та засоби імплементації соціальної взаємодії отримання статичних та динамічних даних	25.10.2023	виконано
4	Моделі та засоби імплементації соціальної взаємодії	22.11.2023	виконано
5	Оцінювання отриманих імплементацій соціальної взаємодії	20.12.2023	виконано
6	Затвердження пояснювальної записки роботи завідувачем кафедри	25.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 83 с., 22 рис., 1 табл., 44 джерела

Тема: Моделі, методи та засоби контролю доступу до статичної та динамічної інформації

Об'єкт дослідження: процеси контролю доступу та захисту інформації.

Мета роботи: розробка ефективних та безпечних механізмів контролю доступу та захисту статичної та динамічної інформації в соціальних мережах для забезпечення конфіденційності, цілісності та доступності інформації, а також контроль над її використанням.

Предмет дослідження: моделі, методи, алгоритми та методологія контролю доступу та захисту інформації в соціальних мережах

Результати дослідження

Особистий внесок студента полягає в розробці та вдосконаленні моделей, методів та засобів контролю доступу до статичної та динамічної інформації які можуть включати нові концепції моделей, алгоритми для реалізації контролю доступу та інноваційні засоби для підвищення ефективності та безпеки інформаційних систем

Висновок

В результаті досліджень було запропоновано метод визначення шкідливих впливів на параметри захисту інформації в соціальних мережах, який, на відміну від існуючих, відрізняється застосуванням системи диференційних рівнянь, які враховують комплексні параметри нападу.

КОМПОЗИЦІЯ, ГЕНЕРАЦІЯ КОДУ, СОЦІАЛЬНІ МАШИНИ, ІНФОРМАЦІЙНІ ЗАГРОЗИ, КРИТЕРІЇ, ДОМЕННО ОРІЄНТОВНА МОВА, ПРОТОКОЛ, ДИНАМІЧНА ІНФОРМАЦІЯ

ABSTRACT

Master's thesis: 83 pp., 22 figures, 1 table, 44 sources

Topic: Models, methods and means of controlling access to static and dynamic information

Object of research: processes of access control and information protection.

The purpose of the work: development of effective and safe mechanisms for access control and protection of static and dynamic information in social networks to ensure confidentiality, integrity and availability of information, as well as control over its use.

Research subject: models, methods, algorithms and methodology of access control and information protection in social networks

Research results

The student's personal contribution consists in the development and improvement of models, methods and tools for controlling access to static and dynamic information, which may include new concepts of models, algorithms for implementing access control and innovative tools for improving the efficiency and security of information systems

Conclusion

As a result of the research, a method for determining the harmful effects on the parameters of information protection in social networks was proposed, which, unlike the existing ones, is distinguished by the use of a system of differential equations that take into account the complex parameters of the attack.

COMPOSITION, CODE GENERATION, SOCIAL MACHINES, INFORMATION THREATS, CRITERIA, DOMAIN ORIENTED LANGUAGE, PROTOCOL, DYNAMIC INFORMATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AJAX – Asynchronous JavaScript and XML

ADL – Architecture Description Language

API – Application Program Interface

CBSE – Component-Based Software Development

CRUD – Create, Read, Update, and Delete

CSS – Cascading Style Sheets

DNS – Domain Name System

DSAL – Domain-Specific Aspect Languages

DSL – Domain-Specific Language

DSML – Domain-Specific Modeling Languages

EMF – Eclipse Modeling Framework

GEF – Graphical Editing Framework (Eclipse)

GMF – Graphical Modeling Framework (Eclipse)

GPL – General-Purpose Language

GQM – Goal Question Metric

GSM – Global System for Mobile communication

GUI – Graphical User Interface

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

IDE – Interactive Development Environment

IaaS – Infrastructure as a Service

IoC – Inversion of Control (IoC),

IPS - Information Processing System

ВСТУП

Актуальність роботи

У сучасному світі інформація потребує надійного захисту: від несанкціонованого доступу і поширення, випадкового видалення або зміни. Робота на тему "Моделі, методи та засоби контролю доступу до статичної та динамічної інформації" залишається дуже актуальною у зв'язку із постійно зростаючою кількістю цифрової інформації та збільшенням загроз кібербезпеки. Забезпечення конфіденційності, цілісності та доступності інформації є важливою задачею для компаній, установ, а також окремих користувачів. Із розвитком Інтернету речей, хмарових технологій та інших інновацій це стає ще більш важливим.

Зростання обсягів інформації: Сучасний світ стикається з величезним обсягом цифрової інформації, що ставить питання ефективного та безпечного контролю за доступом до неї. Забезпечення конфіденційності та безпеки даних стає критично важливим завданням.

Зростання кількості загроз кібербезпеці: Кількість кіберзагроз та атак на інформаційні системи неперервно збільшується. Розробка нових та вдосконалення існуючих методів контролю доступу є важливим елементом у боротьбі з цими загрозами.

Динамічність інформації: З впровадженням динамічних систем та робочих процесів стає важливим ефективний контроль доступу до змінюваної та потокової інформації.

Захист особистої інформації: Особливу увагу приділяють заходам контролю доступу через зростання обігу особистої інформації в електронному вигляді та дотриманням нормативно-правових вимог з питань приватності.

Комплексність інформаційних систем: З ускладненням структури інформаційних систем виникає необхідність вдосконалення методів контролю доступу, які забезпечать адекватний рівень захисту.

З урахуванням цих факторів, дослідження в області контролю доступу до

статичної та динамічної інформації важливе для розвитку безпеки інформаційних систем та відповідності сучасним вимогам кібербезпеки.

Порівняння роботи з відомими розв’язаннями проблеми

Порівняння роботи з відомими розв'язаннями проблеми контролю доступу до статичної та динамічної інформації може включати декілька ключових аспектів: Ефективність, якість контролю доступу, продуктивність: Як швидко та ефективно реалізовані методи обробляють великий обсяг інформації, гнучкість, адаптабельність до різних сценаріїв, сумісність з різними технологіями, безпека, захист від атак, відповідність стандартам безпеки, вартість та співвідношення вартість/ефективність: інтеграція та розширюваність, легкість інтеграції, можливість розширення, використання у реальних умовах, практичність.

Мета і задачі дослідження

Метою магістерської роботи є розробка ефективних та безпечних механізмів контролю доступу та захисту статичної та динамічної інформації в соціальних мережах для забезпечення конфіденційності, цілісності та доступності інформації, а також контроль над її використанням.

Досягнення мети включало розв’язання таких **задач**:

- аналіз існуючих моделей контролю доступу

- вивчення різних підходів та моделей, що використовуються в сучасних системах контролю доступу до інформації.

- Розробка нових методів контролю доступу:

- Розробка та формалізація нових методів, які можуть враховувати сучасні виклики та вимоги безпеки інформації.

- Реалізація розроблених методів у реальній або контрольованій середовищі та проведення тестування їх ефективності.

- Оцінка придатності до практичного використання:

- Визначення можливостей та обмежень запропонованих методів у реальних умовах та їхню відповідність вимогам конкретних застосувань.

Об’єктом дослідження є процеси контролю доступу та захисту інформації.

Предмет дослідження: моделі, методи, алгоритми та методологія контролю доступу та захисту інформації в соціальних мережах.

Методи дослідження

Для досягнення поставленої мети були використані різноманітні методи дослідження, серед яких: аналіз існуючих моделей та методів контролю доступу, емпіричні дослідження, задачі дослідження, аналіз існуючих моделей контролю доступу, розробка нових методів контролю доступу, впровадження та тестування, оцінка придатності до практичного використання.

Наукова новизна отриманих результатів

Отримані результати можуть бути корисні для галузей, де важлива захист інформації, таких як фінанси, медицина, галузі, пов'язані із зберіганням особистої інформації та багато інших сфер. Наукова новизна може виявитися в таких аспектах: Нові моделі контролю доступу, ефективні методи обробки статичної та динамічної інформації, інноваційні засоби контролю доступу, адаптація до нових викликів.

Практичне значення одержаних результатів

Практичне значення отриманих результатів: полягає в забезпеченні безпеки інформації, застосування в індустрії та бізнесі, адаптація до сучасних технологій, захист особистої інформації, використання у сфері кібербезпеки, сприяння розвитку наукових та технічних інновацій.

Особистий внесок студента

Розробка та вдосконалення моделей, методів та засобів контролю доступу до статичної та динамічної інформації. Ці результати можуть включати нові концепції моделей, алгоритми для реалізації контролю доступу та інноваційні засоби для підвищення ефективності та безпеки інформаційних систем.

Структура та обсяг магістерської роботи

Магістерська робота викладена на 83 сторінках друкованого тексту, який складається із вступу, чотирьох розділів, висновків, списку використаних джерел (44 найменування). Робота містить 1 таблицю та 22 рисунки.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПОБУДОВИ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Нові тренди для веб-інформаційних систем

Серед кількох різних доменів один із них переживає значне зростання кількості додатків, що розробляються для веб-домену та в ньому, ймовірно, завдяки зростанню кількості загальнодоступних програмних інтерфейсів (API).

В останні десятиліття ми спостерігаємо перехід від автономного програмного забезпечення до високопідключених систем, головним чином через надзвичайну популярність і всюдисущість Інтернету. На початку комерційного Інтернету Інтернет був, в основному, набором ресурсів «тільки для читання», контент якого створювався декількома спеціалізованими експертами або окремими IT-компаніями та споживався переважно меншою групою користувачів-початківців. У той час (середина 1990-х) поширеною діяльністю був веб-серфінг або анонімне завантаження на сервері FTP (протокол передачі файлів) . Звичайні користувачі споживали вміст і рідко завантажували дані. Зараз такі типи транзакцій все ще часто відбуваються, коли хтось виконує простий веб-пошук у пошуках певного вмісту.

З масовою присутністю людей з’явилася концепція Web 2.0, що дозволяє звичайним користувачам споживати та створювати (читати-записувати) більше інформації. Це час персональних веб-сайтів, веб-журналів (або блогів), багатокористувацьких ігор, соціальних мереж і так далі. Ці дії включають користувачів, які масово створюють і споживають контент. Завантаження та обмін мультимедійним вмістом стає звичайною діяльністю, настільки, що понад 70% Інтернет-трафіку складається з відео, а до 2018 року цей показник становитиме понад 80% 1 . Багато з цих відео створюють звичайні користувачі, які щоденно обмінюються будь-якою інформацією.

За останні роки ми спостерігаємо ще одну велику зміну. Завдяки великій кількості доступних загальнодоступних API Інтернет стає програмованим (Web 3.0). Рис. 1.1 дає нам уявлення про такий сценарій із зростанням веб-інтерфейсів API з 2005 року. ProgrammableWeb є найбільшим у світі репозиторієм API, каталогізуючи понад 10 000 API на кінець 2013 року та досягнувши понад 12 000 API у другій половині 2013 року. 2014. Там можна побачити кількість загальнодоступних API, доступних багатьма компаніями, включаючи великих гравців, таких як Google, Facebook, Amazon, Twitter тощо. Кілька API доступні для різних типів послуг, таких як: картографічні, електронна комерція, соціальні мережі, телефонія, медіа (аудіо/відео) потокове передавання, реклама, пошук, зберігання та багато іншого. У цьому сенсі, створення нової програми на основі Web 3.0 набагато більше стосується написання зв'язків між доступних послуг, ніж створення спеціальної інформаційної системи. На рисунку 1.2 показано зведення найпоширеніших категорій API та їх відповідне зростання з 2009 року.

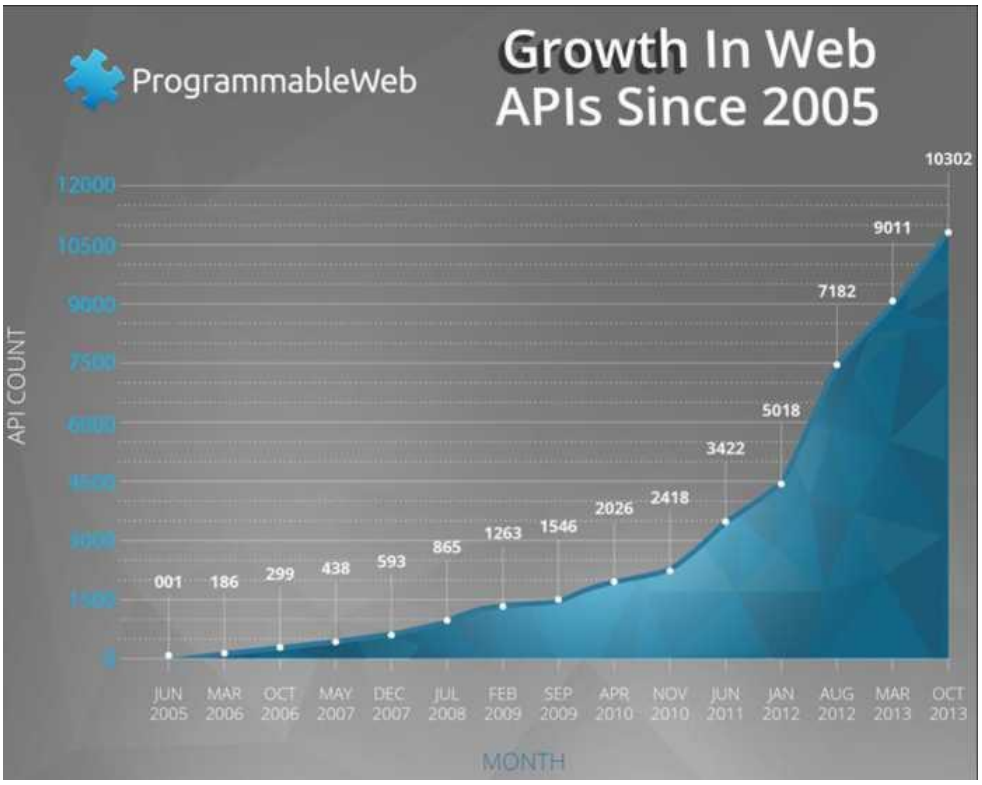

Рис. 1.1. Знімок Інтернету як програмованої платформи зі зростанням веб-API

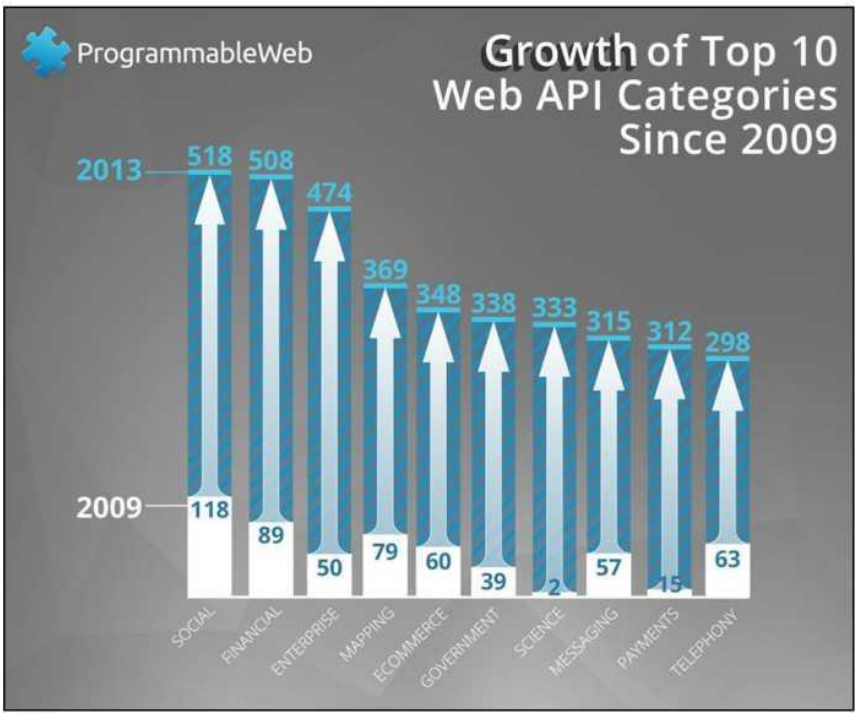

Рис. 1.2. 10 найпоширеніших категорій API та їх відповідне зростання

Незважаючи на досить велике впровадження загальнодоступних API, є деякі проблеми для початку програмування Web 3.0, а саме:

- Різні мови програмування: мови загального призначення зазвичай керуються парадигмою, такою як об’єктна або функціональна. Кожна мова програмування має власні структури, які дозволяють програмісту абстрагувати реальні проблеми в обчислювальному світі, такі як класи/об’єкти, поліморфізм, успадкування, лямбда-вирази та інші. Однак GPL зазвичай не охоплює конкретних концепцій домену, що призводить до необхідності використання додаткових фреймворків та API, додаючи багато шаблонного коду до програмного забезпечення та роблячи його менш зручним для повторного використання та підтримки. Це не відрізняється від веб-домену. Кілька фреймворків на кількох різних мовах програмування були розроблені та використані для створення веб-додатків. Оскільки кожна програма стає веб-основою, використання таких фреймворків стає обов’язковим, що збільшує складність цих програм.

- Різноманітність базових технологій і стандартів: окрім різних мов, які можна вибрати під час розробки веб-програмного забезпечення, існує багато технологій і стандартів, які можна прийняти. Такі технології, як REST, SOAP, JSON-RPC 1 , RSS 1 2 , XML-RPC і навіть SMS (Short Message Service) є досить популярними, коли програмам потрібно обмінюватися інформацією. в Інтернеті. Кожна з них може використовуватися різними мовами програмування по-різному. Наприклад, один певний REST API може бути однаково реалізований у Java (з Java-анотаціями) або в Python з абсолютно іншою структурою, навіть якщо сам API поводитиметься подібно. Крім того, розробник повинен знати, як працює REST і, як наслідок, базовий протокол зв'язку (HTTP), щоб відповідним чином зробити API доступним.
- Відсутність абстрактної моделі для додатків на основі Web 3.0: парадигми програмування зазвичай визначають фундаментальну абстрактну модель, яка допомагає розробникам створювати програмне забезпечення реального світу. Ця абстрактна модель зазвичай дає розуміння для створення більш складного програмного забезпечення без необхідності розуміти багато деталей того, як а комп'ютер працює, наприклад, у сфері електроніки чи фізики. Щоб справлятися зі зростаючою складністю реального світу, мови програмування мали підвищити рівень абстракції. Це те, що відбувається, наприклад, з об'єктно-орієнтованою парадигмою. Абстрактна модель «класів і об'єктів» допомагає впоратися з інкапсуляцією, модульністю, повторним використанням та іншими проблемами, пов'язаними з сучасним програмним забезпеченням. У цьому сенсі однією з основних проблем при створенні додатків Web-3.0 є відсутність абстрактної моделі для опису конкретних проблем, пов'язаних з Web. Цей тип програми, природно, містить концепції від протоколів зв'язку до питань авторизації та безпеки, що значно підвищує рівень абстракції. Однак такі концепції не вбудовані в базову мову програмування, що зобов'язує розробників присвячувати зусилля реалізації та тестуванню конкретних проблем, пов'язаних з Інтернетом, замість того, щоб приділяти увагу основним бізнес-правилам інформаційної системи.

Отже, мета роботи, описаної в даній роботі, можна сформулювати так:

«Для того, щоб краще зрозуміти цю нову Web 3.0, що з'являється, і впоратися з розмаїттям мов програмування та базових технологій, у цій роботі представлено нову фундаментальну абстрактну концепцію під назвою « Соціальні машини» (SM) і мову високого рівня під назвою SMADL («Соціальні машини»). Мова опису архітектури), щоб описати такі мережі, як спроба створити інший спосіб програмування Інтернету, змішуючи концепції мов опису архітектури (ADL) і доменно-специфічних мов (DSL). SMADL прозоро вбудовує такі концепції, як взаємозв'язки додатків та їхні відповідні обмеження, а також процеси авторизації з їхніми проблемами безпеки».

Оскільки запропонована модель соціальної машини та SMADL є частиною ширшого контексту, деякі пов'язані з нею аспекти будуть виключені. Тим не менш, оскільки ці аспекти були передбачені з початкових визначень структури, їх можна буде додати в майбутньому з деякими коригуваннями. Таким чином, наступні питання безпосередньо не розглядаються в цій роботі:

• Система доменних імен (DNS) для соціальних машин: щоб інтегрувати соціальні машини через Інтернет, необхідно, щоб вони виявляли одна одну на основі певного типу опису, ідентифікації чи адреси, подібно до того, що DNS робить із поточним Інтернетом. . DNS перетворює URL-адресу на основі легко запам'ятовуваної послідовності символів в IP-адресі, яка унікально ідентифікує машину на Інтернет. У цьому контексті DNS для соціальних машин перетворить базовий опис, так само як «відображення» або «сховище», на відповідну URL-адресу машини. Це було б більше схоже на пошукову систему та DNS разом. Виникли б інші запитання, якби така система була доступна, наприклад: що, якщо дві машини мають однаковий опис, яка з них має пріоритет для відповіді на запит? Що, якщо одна машина вимкнеться, чи повинна інша подібна машина з подібними послугами замінити її? Ця робота безпосередньо не розглядає жодне з цих питань і не реалізує DNS-подібну систему для соціальних машин. Ми припускаємо, що відносини між СМ встановлюються лише в тому випадку, якщо вони раніше знають один одного.

- SMADL визначається як формальна мова: під час нашого розуміння характеристик DSL і ADL ми зрозуміли, що використання формальної семантики в мові відверне потенційних користувачів, оскільки писати та підтримувати код такою мовою буде складніше. Тоді ми вирішили розробити просту граматику для SMADL без формальної семантики.
- Процес створення мов високого рівня: під час нашого дослідження DSL і ADL ми змогли каталогізувати кілька технік, методів і процесів для створення та підтримки мов високого рівня. Ця робота не призначена для структурування процесу створення DSL або ADL, а для повторного використання існуючих методів для створення SMADL.

В результаті роботи, представленої в цій роботі, можна перерахувати такі внески:

• Представлення абстрактної моделі для Web 3.0: соціальна машина – це нова концепція, яка допомагає зрозуміти складні структури програмованого Web.

• Визначення мови, яка реалізує модель соціальної машини: SMADL було повністю реалізовано та протестовано на основі концептуальної моделі SM, представляючи новий спосіб програмування Інтернету. Крім того, щоб повністю використати переваги високого рівня абстракції SMADL, його було реалізовано у двох версіях, які розширили модель SM: одна текстова та одна візуальна мова, кожна з яких застосовна до різних контекстів, генеруючи різні типи коду та демонструючи широка застосовність самої моделі та мови.

• Визначення, планування, функціонування, аналіз та інтерпретація прикладу з використанням SMADL: після виконання дослідження з використанням текстової версії SMADL ми змогли скоротити загальні зусилля для впровадження програм Web 3.0, згенерувавши частину коду.

- Виконання та інтерпретація опитування з експертами: опитування, проведене з експертами для оцінки візуальної версії SMADL, продемонструвало практичну застосовність мови.

- Реалізація інструменту для обробки візуального синтаксису SMADL: щоб маніпулювати діаграмами, створеними візуальною версією SMADL, було створено графічний інструмент, який використовувався в промислових умовах.

Ця робота повністю описана в усьому документі, а її структура пояснюється в послідовності.

У пошуках інших мов високого рівня, які могли б вирішувати проблеми Web 3.0, ми виконали систематичне дослідження відображення предметно-специфічних мов, що допомогло нам зрозуміти, які DSL були розроблені до цього часу, які домени є ті DSL, до яких застосовуються, і які техніки, методи та процеси використовуються для розробки та підтримки таких мов. Під час виконання цього дослідження ми визначили інші типи досліджень, у тому числі мови опису архітектури (ADL).

Базуючись на концепції SM, у розділі 5 ми пропонуємо мову опису архітектури соціальних машин (SMADL). SMADL реалізований у двох різних версіях: текстовій (t-SMADL) і візуальній (v-SMADL). Кожна версія потім деталізується, і кожен елемент концептуальної моделі SM відображається в мові.

У цьому розділі ми представляємо систематичне картографічне дослідження, щоб краще зрозуміти сферу досліджень DSL, шляхом синтезу доказів, щоб запропонувати важливі наслідки для практики, а також виявлення тенденцій дослідження, відкритих питань і областей для вдосконалення. Картографічне дослідження – це підхід, заснований на фактичних даних, який використовується для того, щоб надати огляд дослідницької галузі та визначити кількість і тип досліджень і результати, доступні в ній. Таким чином, мета цього дослідження полягає в тому, щоб визначити, оцінити та синтезувати сучасні предметно-спеціальні практики програмування, а потім зібрати докази того, що було досягнуто на даний момент у цій дисципліні. Ми також зацікавлені в каталогізації доменів, які скористалися перевагами використання DSL. Таким чином, дослідники та/або практики можуть знати, які DSL були застосовані до конкретного домену, а потім повторно використовувати або адаптувати його для будь-яких інших конкретних потреб.

1.2 Метод дослідження системного картографування отримання даних

MS включає аналіз первинних досліджень, які досліджують аспекти, пов’язані з попередньо визначеними дослідницькими питаннями, спрямовані на інтеграцію та синтез доказів для підтримки або спростування конкретних гіпотез дослідження. MS дозволяє наносити докази в домені на високому рівні деталізації. З іншого боку, систематичний огляд літератури зазвичай виконується для вивчення конкретної галузі дослідження з низьким рівнем деталізації. Якщо під час початкового дослідження домену перед замовленням систематичного огляду буде виявлено, що ймовірно існує дуже мало доказів або що тема є дуже широкою, тоді систематичне картографічне дослідження може бути більш доречним, ніж систематичний огляд. У нашому випадку ми помітили, що тема дослідження, пов’язана з предметно-орієнтованими мовами, дуже широка, з багатьма різними галузями та перетинами з іншими напрямками досліджень. Тоді ми вирішили спершу виконати MS і оцінити реальну потребу у виконанні SR для отримання додаткових деталей.

Основні причини виконання МС можна назвати наступним чином, як це визначено:

- Зробити неупереджену оцінку якомога більшої кількості досліджень, виявити існуючі прогалини в поточних дослідженнях і зробити внесок у наукове співтовариство надійним синтезом даних;
- Забезпечити систематичну процедуру для визначення характеру та обсягу даних емпіричного дослідження, які доступні для відповіді на запитання дослідження;
- Окреслити проведені дослідження;
- Допомогти спланувати нове дослідження, уникаючи непотрібного дублювання зусиль і помилок;
- Визначити прогалини та кластери в ряді первинних досліджень, щоб визначити теми та області для виконання більш повних систематичних оглядів.

Таким чином, ми могли б застосувати процес картографічного дослідження, включаючи найкращі практики для проведення систематичних оглядів, найкраще використовуючи обидва методи.

Весь процес дослідження картографування можна побачити на рисунку 1.3. Він чітко розділений на три основні фази:

1. Директиви щодо дослідження: на цьому етапі визначається протокол дослідження, який складається з дослідницьких питань, стратегії пошуку, джерел даних, які будуть використані, та критеріїв відбору досліджень;

2. Збір даних: цей етап включає виконання MS, під час якого виконується пошук первинних досліджень. Тут розглядається набір раніше визначених критеріїв включення та виключення, які використовуються для вибору досліджень, які можуть містити відповідні результати відповідно до цілей дослідження;

3. Результати: нарешті розроблено схему класифікації та витягнуто дані з документів. Результати ретельного аналізу, проведеного з кожним вибраним первинним дослідженням, повідомляються у формі картографічного дослідження.

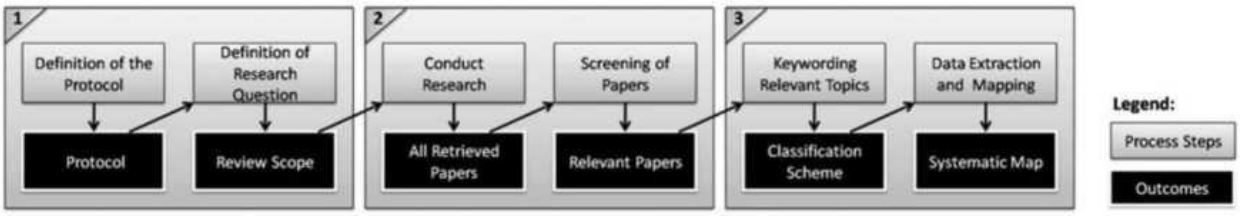

Рис. 1.3.Систематичний процес картографування

1.3 Розвиток мережевих систем та Інтернету

Традиційна концепція програмного забезпечення змінювалася протягом останніх десятиліть. З моменту першого визначення обчислювальної машини, описаного Тьюрингом, програмне забезпечення почало ставати частиною нашого життя та стало повсюдним і повсюдним із появою персональних

комп’ютерів, Інтернету, смартфонів і, нещодавно, Інтернету речей. Насправді можна сказати, що програмне забезпечення та Інтернет змінили спосіб нашого спілкування, спосіб ведення бізнесу, а Інтернет змінює спосіб розробки, розгортання та використання програмного забезпечення. У наш час обчислення означає підключення; і може бути так, що розробка програмного забезпечення – це те саме, що підключення служб.

Ранній Інтернет був мережею здебільшого статичного вмісту, в основному HTML-сторінок, представлених у режимі лише для читання, або систем із дуже простою можливістю транзакцій з точки зору користувача, щось, що можна порівняти з «ящиком» взаємодії пошукової системи); це Інтернет, який ми можемо класифікувати як «1.0».

Як подальший розвиток, з’явилися нові технології, що породило поняття Ajax. Ajax (Asynchronous JavaScript and XML) — це група взаємопов’язаних методів веб-розробки, які використовуються на стороні клієнта для створення асинхронних веб-додатків. За допомогою Ajax веб-програми можуть надсилати дані на сервер і отримувати дані з нього асинхронно (у фоновому режимі), не втручаючись у відображення та поведінку існуючої сторінки. Незважаючи на назву, Ajax не вимагає використання XML (замість нього часто використовується JSON), і запити не обов’язково повинні бути асинхронними. Важливо зауважити, що Ajax — це не одна технологія, а група технологій. HTML і CSS можна використовувати разом для розмітки та стилізації інформації. Це дозволив появі нового типу веб-додатків під назвою: RIA — Rich Internet Application. RIA — це веб-програма, яка має багато характеристик настільного програмного забезпечення, тобто користувачеві дозволено взаємодіяти з програмою браузера так само, як він взаємодіє з програмою настільного комп’ютера.

Відтоді веб-сторінки стали більш інтерактивними та дозволили легко та швидко обмінюватися вмістом, соціальну взаємодію та співпрацю, що призвело до появи блогів, вікі та соціальних мереж. У той час Інтернет розвинувся до «2.0», або читання/запису в Інтернеті, оскільки звичайний користувач зазвичай

завантажує різноманітний вміст, зазвичай мультимедійний (фото, відео, звук). Важливо підкреслити, що кожна фаза Мережі не зникає, коли інша починає діяти. Вони співіснують разом. Звичайний користувач може просто переглядати Інтернет, здійснюючи пошук і отримуючи відповіді, не потребуючи створювати будь-який вміст або активно взаємодіяти з веб-сайтами.

Відтоді за останні роки Інтернет почав значно змінюватися. Зрозуміло, що з'являється нова фаза, Web «3.0», Web як платформа програмування, мережа як інфраструктура для інновацій, на вершині якої всі й усі можуть розпочати розробку, розгортання та надання інформаційних послуг за допомогою обчислювальна, комунікаційна та керуюча інфраструктури в такий спосіб, схожий на такі комунальні послуги, як електроенергія.

У широкому розумінні ми визнаємо, що Web 3.0 – це мережевий простір-час, де інновація полягає в розробці програмного забезпечення для Інтернету через Інтернет і в Інтернеті, використовуючи Інтернет як платформу для програмування та розгортання, і середовище виконання, замінюючи звичайну платформу комп'ютера/операційної системи/середовища розробки » (MEIRA та ін., 2010). Декількома прикладами цього сценарію є поточні розробки у Facebook, Twitter, Yahoo!, Salesforce, Google, Amazon та багатьох інших корпораціях, які роблять свої API доступними для будь-кого для розробки програм, які взаємодіють із їхніми службами. Кожна з таких концепцій, як SOAP, REST, хмарні обчислення та «все як послуга» — XaaS відіграє важливу роль у новому Інтернеті. На рисунку 1.4 узагальнено три фази хвиль Інтернету.

Хоча було проведено багато досліджень про майбутнє Інтернету та таких концепцій, як Web 3.0, програмована мережа, зв'язані дані і семантичної мережі, сегментація даних і проблеми, пов'язані з комунікацією між системами, заплутують інтерпретацію цього майбутнього. Кевін Келлі, відомий Wired, одного разу сказав: «Інтернет — це найнадійніша машина, яку коли-небудь створювали. Він зроблений із недосконалих, ненадійних частин, з'єднаних разом, щоб зробити найнадійнішу річ, яку ми маємо ». Неструктуровані дані, ненадійні частини та проблемні, немасштабовані протоколи – усе це рідні

характеристики Інтернету, який розвивався протягом 40 років; водночас вони є хорошим, поганим і потворним у Мережі, на яку ми все більше й більше покладаємось у повсякденному житті всього, що потребує об'єднавчого погляду та пояснень, щоб розробляти, розгортати та використовувати в ефективнішим і ефективнішим способом.

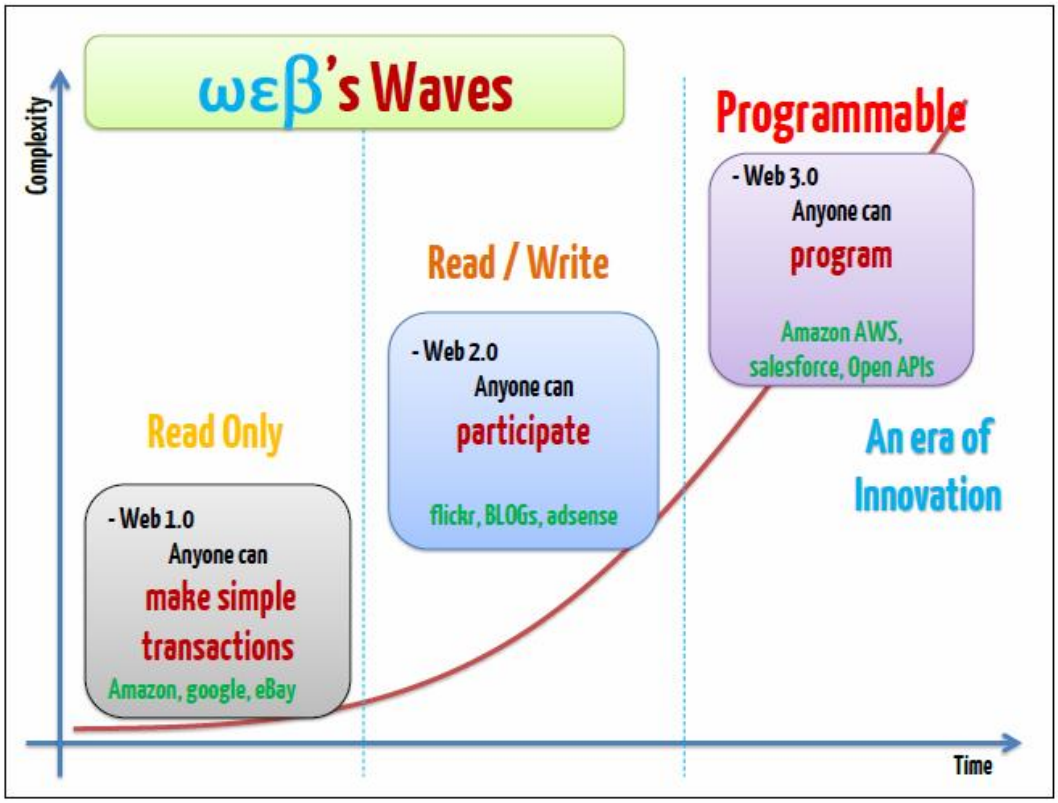

Рис. 1.4. Еволюція веб-концепцій тв Інтернету

Дійсно, Інтернет кардинально змінюється. Однак Інтернет для читання/запису та програмування створені досить недавно, щоб створити дуже серйозні труднощі в розумінні їхніх основних елементів і того, як їх можна ефективно поєднувати для розробки реальних практичних систем в особистому, соціальному чи корпоративному контекстах. Не було а чіткий, точний опис кожної сутності в цьому новому Інтернеті, що виникає, і ми вважаємо, що необхідно створити нові ментальні моделі такого Інтернету як платформи, щоб забезпечити загальну та послідовну концептуальну основу для розуміння цього

молодого, майбутній і, можливо, дуже інноваційний етап розробки програмного забезпечення.

У цій роботі ми досліджуємо нову концепцію під назвою «Соціальні машини» (SM), яка є новим способом, який намагається пояснити цей новий Web 3.0, або програмований Web. Це не теоретична робота, але, більш ніж в одному сенсі, ми віримо, що концепція соціальної машини може співпрацювати в процесі надання об'єднавчого бачення для опису веб-інформації та практичного вирішення складності цієї нової мережі.

1.4 Концепція соціальних машин, як спосіб представлення програмованого Інтернету

Соціальні машини представляють машини, якими керує людина, відповідальна за соціалізацію інформації серед спільнот, тобто перетин сфер і досліджень соціальної поведінки та обчислювальних систем. Соціальні машини в Інтернеті – це процеси, в яких люди виконують творчу роботу, а машина виконує адміністрування, маючи на увазі те, що відбувається з соціальними мережами та їхніми основними соціальними взаємодіями в спільноті друзів.

У сучасній робототехніці існує поняття соціальної машини, але воно також не пов'язане з тим, що тут пропонується; робототехнічний погляд на соціальну машину – це той, який машина може спілкуватися з людьми та/або взаємодіяти з ними, тобто може демонструвати співчуття людям. Але спроби побудувати «машини емпатії» не є новими: японські chahakobi ningyo, заводні ляльки, які здатні подавати чай, існують століття тому і є попередниками всіх видів японських роботів, які так чи інакше взаємодіють з людьми сьогодні.

«Соціальні машини» — це основа для реконтекстуалізації та переоцінки бізнес-вигод пристроїв, підключених до Інтернету («Інтернет речей»), шляхом інтеграції всього, чого ми навчилися в соціальних мережах. Це спосіб переосмислити цінність Інтернету речей, щоб зробити його більш доступним і значущим для всіх. Цей підхід забезпечує нову концептуальну архітектуру для

розробників продуктів, яка допомагає перенести обговорення з суто гіпотетичний до конкретного — надаючи їм чітку та зрозумілу точку зору" Дійсно, визначення дає гарне розуміння СМ. Однак йому бракує одного фундаментального аспекту: «програмованого» аспекту. На нашу думку, кожна соціальна машина повинна бути програмованою та якимось чином повинна надавати свої послуги програмованим, динамічним способом. Його визначення також зосереджується на соціальних мережах, але, незважаючи на саму назву «соціальна» машина, вона не обов'язково має бути пов'язана з соціальними мережами.

Зважаючи на це, дуже ймовірно, що представлене тут поняття соціальних машин деякий час чекало свого опису, тому ми не стверджуємо, що винайшли його, а скоріше відкрили нову інтерпретацію виразу в конкретному налаштування. У цьому сенсі мережа програмованих машин, які з'єднані одна з одною, а також з'єднують людей та установи в мережу обчислень, зв'язку та контролю, потребували набагато більш абстрактного опису та формалізації, ніж лише її зовнішня поведінка у формі публічного (Web) інтерфейс і кількість API на додаток до стандартних Інтернет-протоколів де-факто.

У цьому контексті ми спробуємо визначити мережу програмованих машин у термінах тлумачення, яке ми надаємо вислову «соціальні машини», без жодної надії охопити весь предмет, його ширші наслідки та навіть що більш важливо, її фундаментальні теорії.

Наразі соціальна машина визначається як:

«Підключений і програмований будівельний блок, який огортає (WI) систему обробки інформації (IPS) і визначає набір необхідних (RS) і наданих послуг (PS), динамічно доступних за обмежень (C), які визначаються, серед іншого , його відносини (Rel) з іншими».

Згідно з цим визначенням, таким чином, SM можна розглядати як будівельний блок програмного забезпечення (компонент), який надає набір послуг, які можуть змінюватися відповідно до його «соціальних» відносин з іншими. Бурегіо назвав це «компонентом соціальних послуг » (SSC). Ці

будівельні блоки здатні взаємодіяти один з одним, створюючи нові системи, як показано на рис. 1.5.

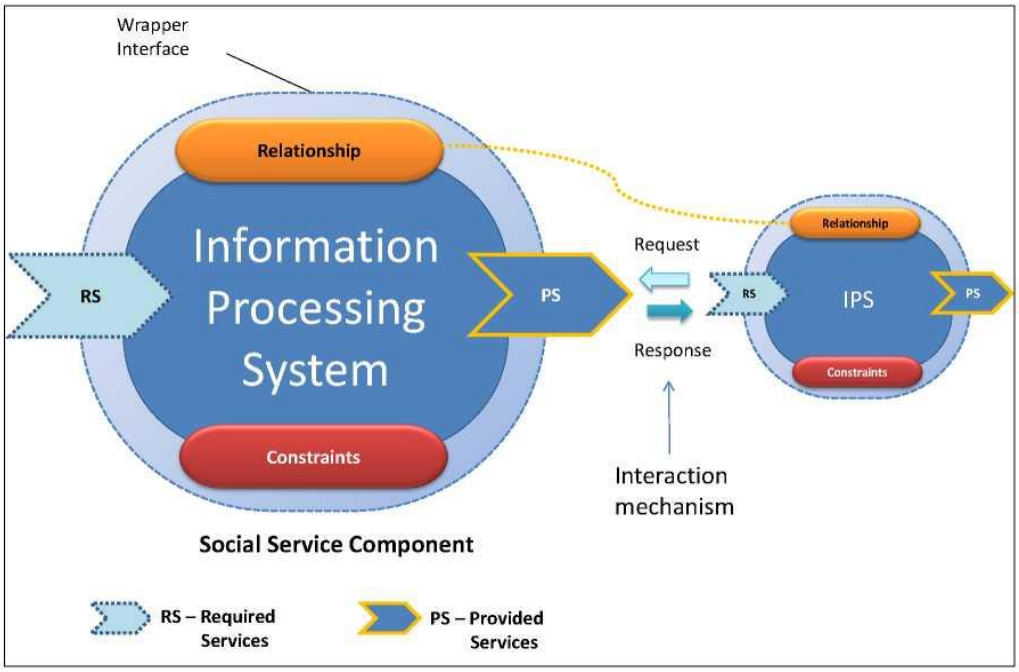

Рис. 1.5. Ілюстрація соціальної машини як компонента соціальних послуг

SM побудований на трьох ключових концепціях: обчислення, зв’язок і контроль. Обчислення є алгоритмічною частиною SM, представленою системами обробки інформації. Поняття комунікації в SM безпосередньо пов’язане з елементами: зв’язок, інтерфейс оболонки, надані та необхідні послуги. Керуюча частина безпосередньо пов'язана з елементом Constraints. Усі виділені елементи в моделі SM детально описані нижче:

• Система обробки інформації (IPS) абстрагує будь-яку обчислювальну одиницю, поведінка якої визначається функціональним співвідношенням між входами та виходами. Це може бути апаратне або програмне забезпечення. IPS може бути представлений, наприклад, алгоритмом, веб-службою або навіть мережею комп’ютерних процесів у різних масштабах або рівнях деталізації. Загалом IPS — це будь-який механізм, який реалізує бізнес-логіку SM.

• Зв'язок (Rel) є центральною частиною моделі SM. Відносини визначають типи взаємодії між обчислювальним процесом і його середовищем,

яке може бути іншою соціальною машиною. На практиці зв’язок між двома соціальними машинами створюється шляхом попереднього встановлення надійного постійного зв’язку між ними. Наприклад, щоб мати певні види взаємодії з програмами, такими як Twitter, Facebook, Dropbox, клієнтська програма повинна бути зареєстрована перед викликом наданих ними послуг, і в більшості випадків різні обмеження пов’язані з цими зв’язками, щоб визначити специфічні види взаємодії. Якщо цей клієнтський додаток раніше не зареєстровано, перегляд взаємодії, тобто динамічна підмножина наданих послуг, буде обмежено лише кількома, зі зниженою швидкістю доступу або обмеженою продуктивністю. Іноді жодна з наданих послуг навіть недоступна. Концепція стосунків між СМ подібна до стосунків між людьми: ми можемо розглядати їх як довірчі відносини між різними СМ, що задовольняють встановлені обмеження, накладені суспільством.

• Інтерфейс оболонки (WI) абстрагує будь-який комунікаційний рівень, через який SM виводить свої служби назовні, щоб дозволити взаємодію з іншими SM, як шаблон проектування fayade. Наприклад, розглядаючи Dropbox як SM, API, який він надає, можна вважати різновидом інтерфейсу-огортки. Через API Dropbox клієнтська програма може взаємодіяти зі своїми основними службами (наприклад, завантаження, завантаження, список, спільний доступ). WI також може нести відповідальність за формування поглядів взаємодії SM відповідно до існуючих обмежень і відносин, які SM має з іншими.

• Надані послуги (PS) представляють бізнес-логіку SM, яка представлена як динамічний набір послуг. Чи цей контекст динамічний означає, що він може змінюватися відповідно до типу відносин, які поточний SM має з запитувачем служби. Загалом надані послуги можна розділити на два основні типи:

- Відкриті загальні служби: представляють загальнодоступні служби, доступ до яких не вимагає попереднього встановлення певних відносин між постачальником SM і його клієнтськими програмами. На практиці відкриті служби приймають запити від «невідомих» і неавтентифікованих програм.

Yahoo та Google Maps є гарними прикладами провайдерів, які пропонують досить відкриті послуги, доступні будь-якій програмі через їхні публічні API. Беручи нашу аналогію з людськими стосунками, відкриті служби можуть представляти, наприклад, загальний набір взаємодій, які ми встановлюємо для незнайомих або незнайомих людей, наприклад, звичайне «доброго ранку». У випадку СМ цей набір послуг утворює єдине і загальне подання взаємодії, доступ до якого обмежений загальним набором обмежень. Як приклад, соціальна машина може обмежити кількість запитів від певної IP-адреси до своїх відкритих служб. Dropbox, наприклад, накладає обмеження на пропускну здатність своїх API, коли до нього звертаються з програми, відмінної від їх власної. Twitter накладає обмеження на швидкість своїх публічних API, а також Google, Yahoo, Facebook та інших.

- Сервіси, керовані зв'язками, окрім відкритих загальних сервісів, SM може зробити доступною групу служб, доступ до яких обмежений відповідно до зв'язків, встановлених між SM та клієнтськими програмами, які з ним взаємодіють. Це випадок SM, який динамічно забезпечує різні види взаємодії відповідно до своїх відносин з іншими. API Twitter, наприклад, відкритий для пошуку загальнодоступної інформації про користувача, але інші операції та умови вимагають попереднього встановлення зв'язку між Twitter і програмою, призначеною для виклику його служб. Кожен набір різних служб представляє конкретне подання взаємодії, але різні подання взаємодії також можуть бути створені одним набором служб, якщо вони надаються з різними атрибутами якості.

• Необхідні послуги (RS). Це необов'язковий елемент, визначений моделлю. Він представляє набір служб, які соціальна машина повинна викликати для належної роботи. Ця концепція може бути корисною при специфікації та аналізі функціональних і структурних залежностей між СМ.

• обмежень (С) визначає правила або обмеження, які мають місце під час встановлення зв'язків і визначення поглядів взаємодії між різними SM.

Існують три різні типи обмежень:

- Обмеження на основі функцій: враховуючи, що кожне представлення взаємодії має дійсну підмножину наданих послуг, існують обмеження на основі функцій, щоб забезпечити дійсну підмножину цих послуг, подібно до того, що відбувається в лінійках програмних продуктів. У цьому контексті ці обмеження гарантують, наприклад, що не повинно бути можливо мати дві альтернативні служби (тобто ексклюзивні (XOR) служби) в одному поданні взаємодії.

- Обмеження видимості, такого роду обмеження пов’язані з обмеженнями видимості послуг, які надає соціальна машина. Вони подібні до модифікаторів доступу в об'єктно-орієнтованих мовах.

- Обмеження якості : цей тип обмежень впливає на атрибути якості послуг, що надаються в поданні взаємодії. Вони можуть визначати, наприклад, протоколи авторизації (для безпеки), кількість запитів за період часу (для продуктивності) або будь-які додаткові властивості, які можуть впливати на будь-який інший атрибут якості. Прикладом цього є Google Maps API, який дозволяє будь-якому незареєстрованому користувачеві здійснювати обмежену кількість дзвінків на день. Якщо клієнтській програмі потрібно більше, ніж це, необхідно встановити відносини (з укладанням договору та оплати), що забезпечує відповідний доступ до API.

Висновки до розділу

У цьому розділі представлено перший етап процесу дослідження картографування, на якому визначено протокол, питання дослідження та критерії відбору досліджень. Виконано аналіз предметної області побудови інформаційних систем, представлені нові тренди для веб-інформаційних систем. Розглянута концепція соціальних машин, як спосіб представлення програмованого Інтернету.

РОЗДІЛ 2
МЕТОДИ ТА ЗАСОБИ ІМПЛЕМЕНТАЦІЇ СОЦІАЛЬНОЇ ВЗАЄМОДІЇ ОТРИМАННЯ СТАТИЧНИХ ТА ДИНАМІЧНИХ ДАНИХ

2.1 Дослідження основних характеристики соціальних машин

Після представлення схематичної концепції SM ми тепер можемо висвітлити деякі з його основних характеристик, а саме:

- Комунікабельність: за самою природою концепції, яку ми пропонуємо, SM є товариськими людьми, і майже в усіх випадках кожен SM повинен надавати засоби для взаємодії з іншим. Ізольована аутична соціальна машина є винятком. SM є не лише можливою основою для опису спільного використання та повторного використання мережевих інформаційних систем, але й має бути реалізовано таким чином, щоб забезпечити це. Ідея, що лежить в основі соціальних машин, полягає в тому, щоб скористатися мережевим середовищем, яке вони вбудовані, щоб полегшити поєднання та повторне використання вихідних служб від різних SM для реалізації нових.
- Композиційність: окрім базових SM «комбінаторів», будь-які SM вищої складності можуть бути представлені в термінах інших SM, що мають меншу складність та/або менше «соціальних навичок». Це дозволяє описувати та впроваджувати SM з використанням підходу «розділяй, щоб володарювати».
- Незалежність від платформи: соціальні машини не залежать від платформи та технологій. На найвищому рівні абстракції ми можемо думати про підхід, запропонований тут, як про новий спосіб опису архітектури інформаційних систем (на базі Інтернету, з інтенсивним використанням Інтернету).
- Незалежність впровадження: SM має надавати свої послуги таким чином, щоб інші SM, які користуються такими послугами, не мали піклуватися про те, як вони були реалізовані. Крім того, бажано, щоб SM використовували чітко визначені та де-факто стандартні протоколи, щоб зв’язок між ними був

максимально простим. Інтерфейс оболонки SM має бути зрозумілим і простим у використанні; тим чіткіше обгортка Інтерфейс полягає в тому, щоб забезпечити доступ до своїх послуг, тим легше іншим буде користуватися їхніми послугами.

• Самосвідомість: кожна соціальна машина повинна бути самосвідомою, за винятком «аутистів», тих, які не підтримують стосунки з іншими і, отже, не потребують нічого знати про себе. Це означає, що СМ повинен мати можливість відповісти на запит «Хто ти?» з відповіддю на кшталт: «Я скорочую URL-адреси, і ви можете користуватися моїми послугами безкоштовно». Ця відповідь містить будь-яку необхідну метаінформацію, наприклад, витрати на використання, доступність, питання конфіденційності тощо.

• Відкриваємість: однією з бажаних характеристик SM є здатність динамічно виявляти інші SM і підключатися до них. Ця специфічна характеристика SM потребує центрального репозиторію для реєстрації будь-якого SM, який, імовірно, буде виявлено, як це робить система доменних імен (DNS) в Інтернеті.

2.2 Типи та приклади соціальних машин як джерел отримання статичної і динамічної інформації

З багатьох можливих класифікацій один із способів поглянути на соціальні машини — це проста таксономія на рис. 2.1, заснована на типах взаємодії, які вони мають одна з одною, які далі описані наступним чином:

• Ізольовані — соціальні машини, які не взаємодіють з іншими соціальними машинами;

• Постачальник — соціальні машини, які надають послуги для інших соціальних машин;

• Споживач — соціальні машини, які споживають послуги, які надають інші соціальні машини;

• Prosumer — соціальні машини, які одночасно надають і споживають послуги.

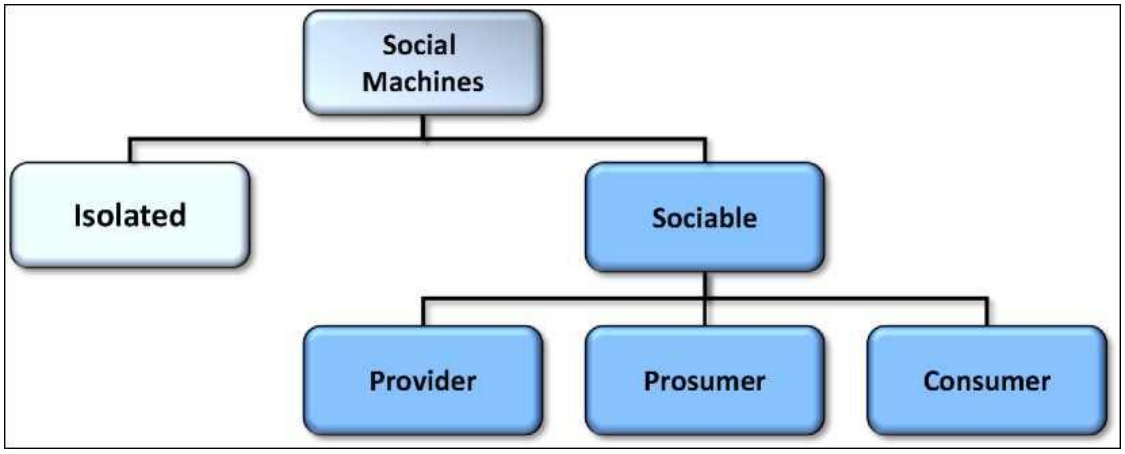

Рис. 2.1. Таксономія соціальних машин

Інтерфейс-оболонка соціальної машини — це елемент, відповідальний за вхідний і вихідний трафік інформації, а отже, інтерфейс для сервісів обміну між соціальними машинами. Якщо дана соціальна машина не надає засобів для взаємодії зі своїми службами, тоді ця соціальна машина класифікується як ізольована. Типовим прикладом є проста автономна програма.

Якщо соціальна машина має здатність взаємодіяти з іншими машинами, то вона називається комунікабельною. Соціальну машину, яка надає послуги іншій соціальній машині, можна класифікувати як постачальника. Такі програми, як Bit.ly, Twitter і Google Maps, є хорошими прикладами соціальних машин як провайдерів.

Соціальна машина також може використовувати послуги, які надають інші. Веб-програми, які споживають послуги з інших веб-програм (зазвичай звані mashup), є прикладами соціальних машин як споживачів. Seesmic, Tweetdeck, HousingMaps і WikipediaVision використовують служби Twitter, Google Maps і Wikipedia.

Нарешті, соціальна машина може як надавати, так і споживати послуги, будучи класифікованою як споживачі. Вони споживають інформацію з однієї чи кількох соціальних машин, обробляють і комбінують цю інформацію та надають

оброблені контекстуалізовані дані іншим соціальних машинам. Гарним прикладом соціальної машини для просумерів є IFTTT. IFTTT отримує доступ до кількох різних популярних веб-інтерфейсів API для створення індивідуальних «рецептів» на основі простих правил, таких як: «якщо зображення завантажено до мого облікового запису Facebook, то збережіть його на моєму хмарному диску». Щоб це стало можливим, IFTTT активує канали (наприклад, Facebook, GMail, Dropbox) і дії, пов'язані з цими каналами (наприклад, «отримати електронний лист», «змінити статус Facebook», «завантажити новий файл»). Таким чином, поєднання різних каналів і дій створює нові сервіси для інших програм і сервісів, і тому IFTTT є хорошим прикладом соціальної машини для споживачів.

Щоб допомогти порівняти ці різні типи соціальних машин, на рисунку 2.4 показана часткова діаграма порядку SM відповідно до їх складності реалізації. Важливо зауважити, що соціальна машина — це динамічна сутність, яка може поводитися лише як споживач, постачальник або просумер у будь-який заданий період часу. На рис. 2.2 показано зв'язки між різними типами SM у вигляді двосторонньої стрілки, чітко показуючи, що SM може змінювати свій тип протягом свого життєвого циклу.

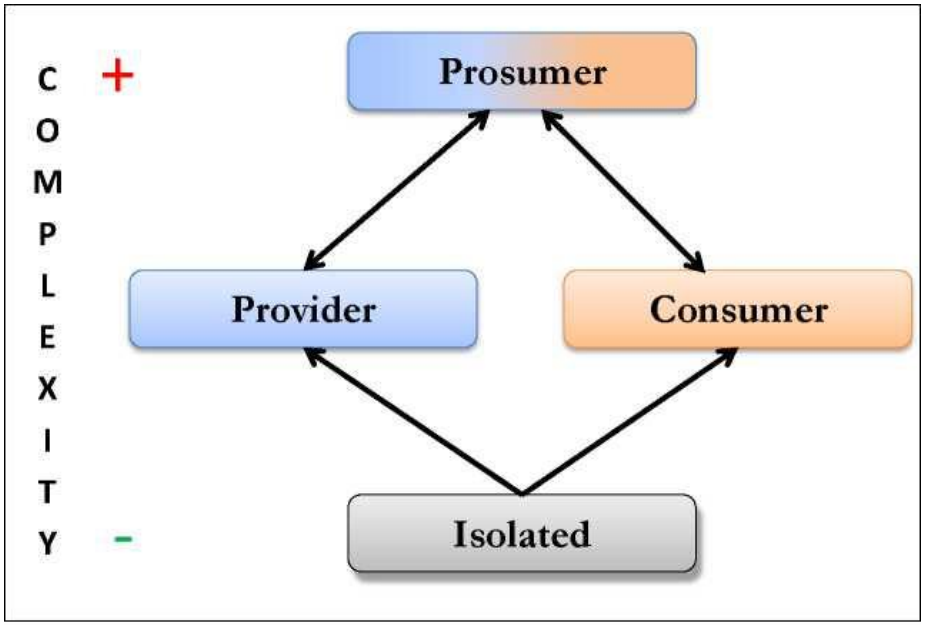

Рис. 2.2. Зв'язки між різними типами SM

Можна згадати кілька прикладів інформаційних систем, які можна вважати соціальними машинами згідно з нашим визначенням. Далі ми представляємо два з них.

Щоб проілюструвати модель SM, ми описуємо тут реальну систему — Futweet, яка була розроблена з використанням об'єднуючої ідеї, яку ми обговорювали досі. Futweet — це і соціальна мережа, і гра в вгадування футбольних (футбольних) результатів. Спочатку Futweet був розроблений для користувачів Twitter, згодом був пов'язаний з іншими онлайн-соціальними мережами, наприклад Facebook і Orkut, що робить його гарним практичним прикладом для ілюстрації розробки програми, яка використовує концепцію Social Machine. Гра ілюструє розробку справжньої соціальної машини для споживачів, оскільки вона була розроблена та створена для об'єднання в мережу з іншими програмами та сама є точкою з'єднання інших програм і служб. На рис. 2.3 ми показуємо всі соціальні машини, які складають соціальну машину Futweet, і її зв'язки.

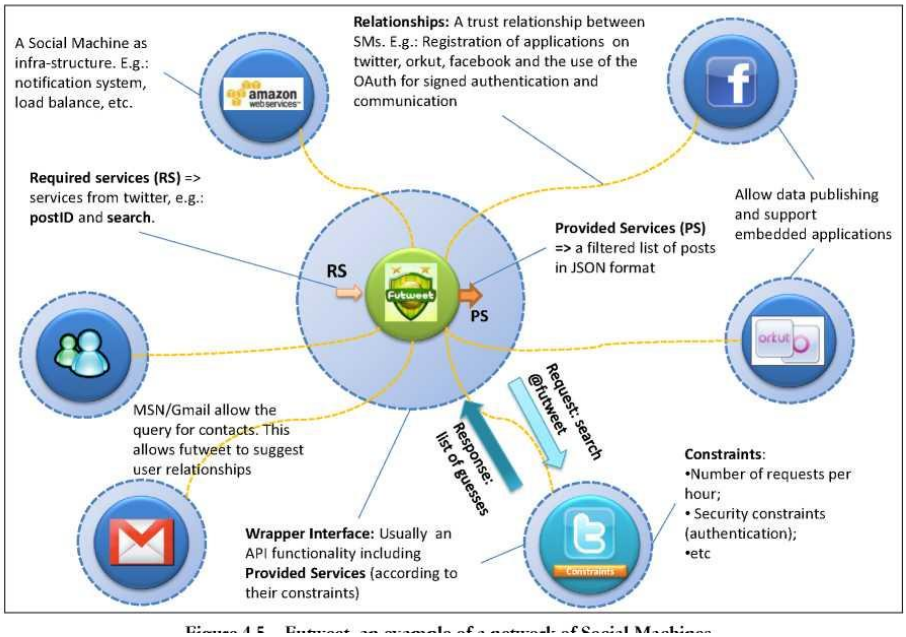

Рис. 2.3. Соціальні машини, які складають соціальну машину Futweet

Ми можемо визначити такі основні елементи соціальної машини у Futweet SM:

- Відносини: Futweet підтримує зв’язки з онлайн-соціальними мережами та іншими SM (наприклад, Amazon EC2). Ці відносини визначають сервіси, які можна вважати частиною інфраструктури соціальних ігор. Якщо якесь із цих SMS недоступне, це може вплинути на Futweet загалом.
- Надані послуги: відфільтрований список публікацій у форматі JSON і поточний рейтинг гравців.
- Система обробки інформації: складається з бізнес-правил Futweet у поєднанні з відповідними програмами в соціальних мережах Facebook і Orkutj, а також механізмами взаємодії з Twitter, Gmail і MSN.
- Необхідні послуги: послуги, які запитує Twitter, як-от пошук дописів, які відповідають заданому шаблону.
- Інтерфейс оболонки: він представлений API Futweet, який інкапсулює основні функції гри, доступні в Інтернеті.
- Обмеження: Futweet має багато обмежень. Деякі з них схожі на обмеження швидкості Twitter API, наприклад кількість записів, які повертає API (дописи, припущення, користувачі тощо). Futweet також обмежує запити для облікового запису та IP-адреси.

Основний механізм гри передбачає надсилання припущень щодо футбольних матчів у певній лізі; такі припущення обробляються та порівнюються з набором попередньо встановлених правил підрахунку очок, і переможцем у грі стає користувач, який отримує більше балів у кінці визначеного періоду, який, як правило, збігається з закінченням відповідного чемпіонату. У випадку з Twitter надсилання прогнозів на матч відбувається за попередньо визначеним синтаксисом, який містить акроніми команди та прогнозовані результати.

Futweet має механізм, який періодично шукає твіти з цим шаблоном, витягує інформацію, яка представляє припущення користувача, а потім перераховує загальний рейтинг. Оскільки Futweet також існує як вбудовані програми на Facebook і Orkut, користувач може запитувати дані (наприклад,

рейтинговий список) у Futweet, використовуючи програми на вершині цих соціальних машин.

Під час розробки Futweet концепція SM допомогла зрозуміти різні типи встановлених зв'язків і створити чистий інтерфейс оболонки RESTful. Оскільки це був перший іграшковий проект, реалізований на основі моделі SM, Futweet показав, що навіть неможливо реалізувати соціальні машини за допомогою визначеної тут концепції, але також легше зрозуміти архітектуру програми та взаємозв'язки.

Елементи соціальної машини можна відобразити в системі Ushahidi наступним чином:

- Відносини: Ushahidi підтримує зв'язки з будь-яким пристроєм із підтримкою SMS і картографічними системами для відстеження інформації про катастрофи. Також можна створювати нові відносини за допомогою служби текстових повідомлень, такої як Facebook або Twitter.
- Надані послуги: повний список місць стихійних лих у регіоні з інформацією, що відстежується з мобільних пристроїв. Будь-хто, хто хоче, може відновити цю інформацію та використати її для порівняння з будь-якими іншими регіонами.
- Система обробки інформації: це здатність Ushahidi обробляти неструктуровані текстові повідомлення від кількох людей із зазначенням їхнього місцезнаходження та впорядковувати їх на відповідній карті.
- Необхідні послуги: послуги запитуються від картографічного сервера для відновлення відповідної карти певного регіону відповідно до інформації, наданої користувачами системи.
- Інтерфейс оболонки: він представлений API Ushahidi, який інкапсулює основні функції отримання текстових повідомлень від пристроїв, що підтримують SMS.
- Обмеження: Ushahidi має певні обмеження, головним чином пов'язані з обмеженням швидкості API картографічної служби, тобто кількістю пошуків будь-якої позиції за хвилину в картографічній службі.

І, нарешті, в SCA провід — це абстрактне представлення з’єднання між посиланням і деякою службою, яка відповідає потребам цього посилання. Проводи можуть проходити через перетворення, щоб забезпечити з’єднання між різними типами послуг, що називається просуванням у контексті SCA. Рис. 2.4 ілюструє композит SCA та його концепції в детальній перспективі. Для кращого розуміння на рис. 2.5 показано загальну картину всіх концепцій SCA та їхніх взаємозв’язків.

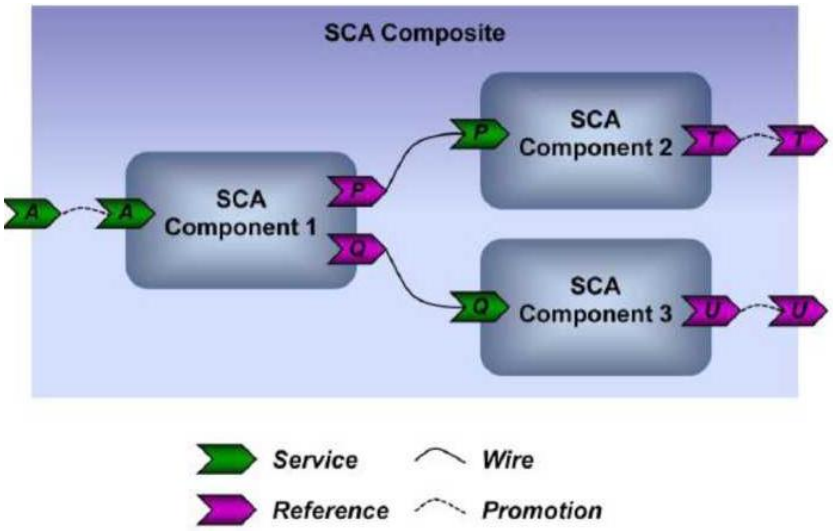

Рис. 2.4. Ілюстрація композиту SCA

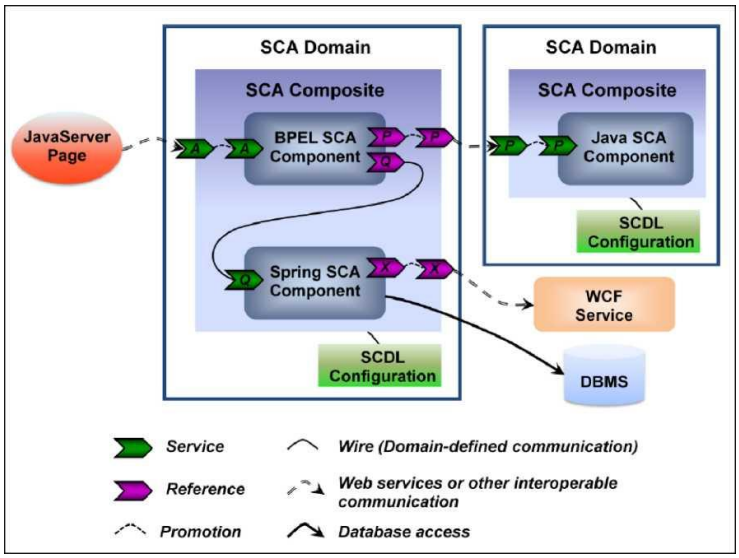

Рис. 2.5. Приклад повного сценарію SCA з його елементами та їх взаємозв’язками

У багатьох аспектах стандарт SCA можна порівняти з концептуальною моделлю SM. Щоб зробити відмінності більш зрозумілими, кожна концепція SCA порівнюється з концептуальною моделлю соціальної машини в таблиці 2.1.

Таблиця 2.1.

Порівняння концепцій SCA та SM

Topics	SCA	SM
Purpose	• Programming / abstract model • Way to describe **compositions** of components using SOA principles	• Programming / abstract model • Architectural style • Design methodology focused on **social aspects** of software
Building blocks	SCA Component	Social Machine
Main elements	• Reference → • Relationships • Service → • Wrapper Interface • Wire → • Information Processing System (input/output) • Promotion → • Request • Binding → • Response • Domain → • Constraints • Composite → • SM network • SCA Runtime → • Internal states and constraint of IPS • Policies	
Technology independency	YES	YES
Runtime environment	Vendor's Specific Runtime	Not required (SMADL is defined on top of JVM)
Keep track of relationship history	NO	YES
Dynamic aspect	Inversion of Control (IoC), in other words dependency injection though references	Desirable (DNS for SMs is out of scope)
IDE	Some open and commercial tools; Ex.: SOA Tools Platform Project	Eclipse-based IDE implemented for SMADL versions
Way to define compositions	Configuration File (SCDL)	Social Machine Architecture Description Language (SMADL)

Іншими відповідними підходами, які можна порівняти із соціальними машинами, є RESTFul Web Services і похідні RESTFul Web API. Однією з ключових характеристик веб-сервісу RESTful є явне використання методів HTTP у спосіб, який відповідає протоколу, визначеному RFC 2616. Наприклад, HTTP

GET визначається як метод створення даних, призначений для використання клієнтська програма для отримання ресурсу, отримання даних з веб-сервера або виконання запиту з очікуванням, що веб-сервер шукатиме та відповідатиме набором відповідних ресурсів. REST просить розробників використовувати методи HTTP явно та у спосіб, який узгоджується з визначенням протоколу. Цей основний принцип розробки REST встановлює однозначне відображення між операціями створення, читання, оновлення та видалення (CRUD) і методами HTTP. Відповідно до цього відображення:

1. Щоб створити ресурс на сервері, використовуйте POST.
2. Щоб отримати ресурс, використовуйте GET.
3. Щоб змінити стан ресурсу або оновити його, використовуйте PUT.
4. Щоб видалити або видалити ресурс, використовуйте DELETE.

Прикрий недолік дизайну, притаманний багатьом веб-API, полягає у використанні методів HTTP для ненавмисних цілей. URI запиту в запиті HTTP GET, наприклад, зазвичай ідентифікує один конкретний ресурс. Або рядок запиту в URI запиту містить набір параметрів, які визначають критерії пошуку, які використовує сервер для пошуку набору відповідних ресурсів. Принаймні так HTTP/1.1 RFC описує GET. Але є багато випадків непривабливих веб-інтерфейсів API, які використовують HTTP GET для ініціювання транзакцій на сервері, наприклад, для додавання записів до бази даних. У цих випадках URI запиту GET не використовується належним чином або принаймні не використовується повністю REST. Підсумовуючи порядок щоб веб-сервіс або веб-API вважався RESTful, він повинен суворо відповідати чотирьом зіставленням між операціями CRUD і методами HTTP, як згадувалося раніше.

У новішій роботі намагається розвинути концепцію REST. Він називається репрезентативним перенесенням стану дії. Ідея базується на ресурсно-орієнтованій веб-автоматизації (ROWA), де мета дизайну полягає в тому, щоб забезпечити виконання веб-завдань таким чином, щоб людина була незамінною. Ця мета дизайну реалізується шляхом ретельного розширення поточного архітектурного стилю Інтернету RESTful. Автор стверджує, що результати цієї

роботи просувають веб-користувачів від поточного стану веб-операторів, які когнітивно обтяжені ручною природою операцій у вебі, до веб-контролерів, чиє когнітивне навантаження під час користування мережею значно зменшується до моніторинг автоматизованих веб-завдань. Робота здається багатообіцяючою, але поки що на початковій стадії.

Хоча веб-сервіси RESTful є важливим принципом проектування, якого слід дотримуватися, насправді вони знаходяться на нижчому рівні абстракції соціальної машини. SM можна реалізувати за допомогою, наприклад, служб RESTful або будь-якої іншої технології. Наша ідея з моделлю SM полягає в тому, щоб визначити зв'язані, комунікабельні організації та їхні відповідні стосунки, не турбуючись про основну технологію.

Концепція соціальних машин перекриває інші галузі досліджень і проблеми, які зараз добре вивчені, такі як SaaS, хмарні обчислення, SOA та соціальні мережі, але, незважаючи на цей факт, ми не знайшли жодного дослідження, яке стосується цієї концепції, як ми пропонуємо тут.

У цьому розділі представлена наша фундаментальна теорія соціальних машин. Щоб синтезувати соціальну машину, це «з'єднуваний і програмований будівельний блок, який огортає (WI) систему обробки інформації (IPS) і визначає набір необхідних (RS) і наданих послуг (PS), динамічно доступних за обмежень (C), які є визначається, серед іншого, його відносинами (Rel) з іншими». Кожен із цих елементів було пояснено та детально описано в цьому розділі. У цій главі ми перерахували деякі загальні характеристики SM і надали кілька прикладів реальних систем, які можна вважати SM. Ми також надали просту таксономію для класифікації різних типів соціальних машин.

Ми каталогізували кілька технік, методів і процесів для маніпулювання мовами, а також ми зрозуміли, в яких доменах вони використовуються. Ми перерахували та порівняли основні мови високого рівня, які використовуються для опису архітектур програмного забезпечення, деякі з них є загальними, а інші – предметними.

Після ознайомлення з концепцією соціальних машин у попередньому розділі та під час усього цього дослідження ми помітили, що багато зусиль витрачається на розробку ефективних способів впровадження інформаційних систем на основі Web 3.0, які повністю відповідають цій новій епосі програмованих Інтернет.

Одним з ефективних способів закріпити фундаментальні концепції певної області, наприклад Web 3.0, у практичних системах є впровадження мови високого рівня, яка вбудовує такі концепції в простий і зрозумілий синтаксис (візуальний або текстовий). Оскільки ми охопили фундаментальні концепції Web 3.0 в абстрактній моделі соціальної машини та ґрунтуючись на наших попередніх дослідженнях DSL та ADL, ми пропонуємо нову мову під назвою SMADL, мову опису архітектури соціальних машин, щоб бути новим способом програмування Інтернету. Коротше кажучи, SMADL можна визначити як мову, що керується взаємозв'язками, яку можна використовувати для опису взаємодії між будь-якою кількістю машин багатьма способами, як засіб представлення реальних машин, що взаємодіють у реальному Інтернеті, наприклад Twitter. які працюють на основі Amazon AWS, мешапів, створених за допомогою Google Maps, ігор, що працюють на основі Facebook, і, очевидно, також як засіб представлення взаємодії з іншими соціальними машинами. Крім того, мова може описувати композиції SM, такі як інфраструктури центрів обробки даних на основі технологій віртуалізації. Це означає, що SMADL можна використовувати для опису не лише програм, що працюють на основі Amazon AWS, але й інфраструктури віртуалізації самої Amazon, наприклад.

2.3 Визначення мови для опису архітектури інформаційних систем на основі Web 3.0

Коли ми вперше задумалися про створення нової мови, першим важким рішенням, яке ми повинні були прийняти, було те, чи буде це текстова чи візуальна мова. Тим не менш, багато авторів візуального програмування

стверджують саме так: графічні представлення кращі просто тому, що вони графічні. Але і графіка, і текст мають свої можливості та обмеження. Програміст більше схожий на читача технічного посібника, ніж на глядача картини: усвідомлений читач, цілеспрямований і керований гіпотезами. Отже, чому графічні представлення такі привабливі, хоча в деяких випадках з графічними представленнями може бути важче працювати та вони забезпечують нижчу продуктивність, ніж текстові?

Різниця між «текстовим» і «чисто графічним» є компромісом між «описовим» і «аналоговим» представленням. Текст, описове представлення, отримує точність вираження з невеликого фіксованого словника, і досягає діапазону вираження шляхом регулярного поєднання елементів словника. Подібним чином читачі вивчають правила інтерпретації, так що звичайний текст читається послідовно (хоча до нього можна отримати довільний доступ), а текст легко впорядковувати та шукати. Звичайний текст не залежить від реакцій сприйняття, характерних для сенсорного режиму; текст легко перекладається з візуального в інші режими, наприклад, читаючи вголос.

З іншого боку, чиста графіка, аналогове представлення, виграє від відображення перцептивних сигналів на інформацію (наприклад, асоціація кольору з типом або розміру з числом). Він може спиратися на необмежений словниковий запас. Графіка може виграти від інформативного враження від цілого, яке забезпечує розуміння структури, але їй бракує точності тексту, оскільки багато інформації є внутрішньою частиною аналогових відображень. Правила тлумачення не так чітко визначені, як для тексту, тому графіка може страждати від неоднозначності тлумачення. Пропонуючи кілька підказок для навігації, графіка вимагає від читача визначення відповідної стратегії огляду.

Таким чином, кожен підхід, графічний чи текстовий, має свої переваги та недоліки. Надихнувшись цим, ми вирішили впровадити SMADL, взявши користь від обох світів.

Ми визначаємо SMADL — мову опису архітектури соціальних машин — як спробу створити зовсім інший спосіб програмування Інтернету, змішуючи

концепції ADL і DSL. Як ADL, він дозволяє описувати соціальні машини (та їх мережі) у термінах зв'язків як абстракції високого рівня, без необхідності вказувати деталі зв'язку (протоколів) та/або методів автентифікації. Як DSL, він дозволяє впроваджувати та інтегрувати веб-сервіси за допомогою динамічно введеного синтаксису, повністю інтегрованого у віртуальну машину Java та IDE Eclipse. Крім того, якщо необхідно, вирази Java і попередньо визначені класи Java можна використовувати як варіант для реалізації деталей певної соціальної машини.

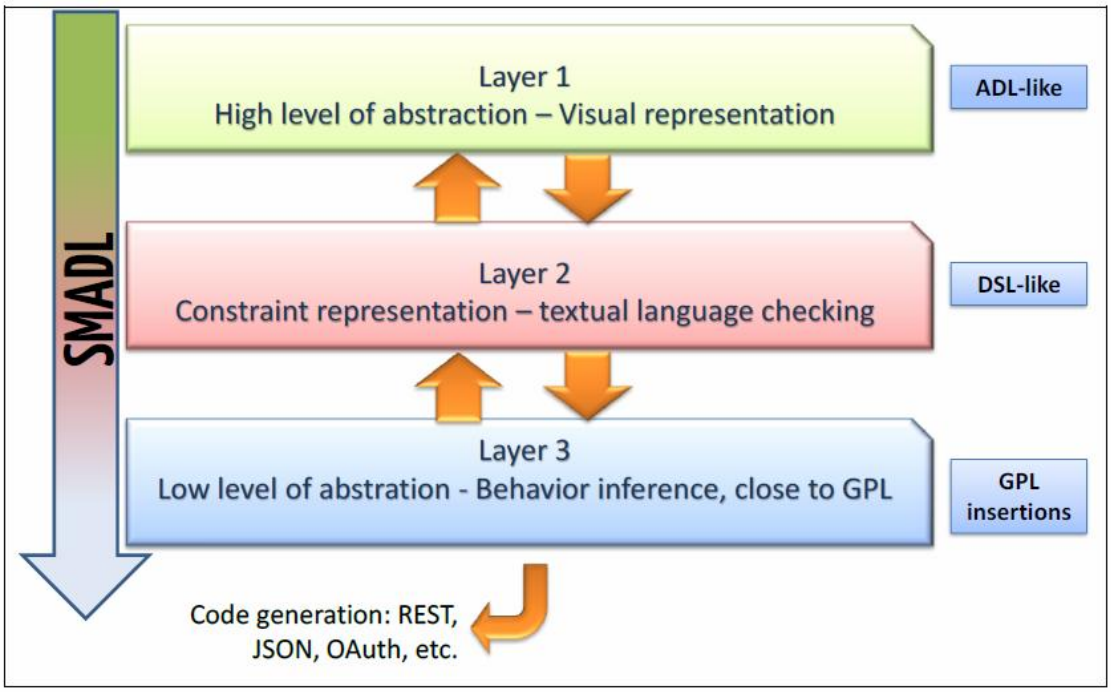

Рис. 2.6. Візуальне представлення SMADL

SMADL реалізовано з використанням різних рівнів абстракції, як показано на рис. 2.6. На рівні 1 візуальне представлення SMADL надається на вищому рівні абстракції. У цьому випадку мова буде виглядати як ADL, вбудовуючи знання предметної області в деякі графічні елементи. На рівні 2 реалізовано текстове представлення SMADL. На цьому рівні абстракції можна перевірити обмеження та граматичні правила, і мова буде виглядати як DSL, з власними попередньо визначеними термінами та виразами, які суворо дотримуються її

синтаксису. I рівень 1, і рівень 2 сумісні з концептуальною моделлю соціальної машини, яка була показана в попередньому розділі.

Нижчим рівнем абстракції є рівень 3, де деякі перетворення застосовуються до моделей рівня 1 і 2, щоб створити та/або виконати код на мові загального призначення, наприклад Java, включаючи REST і OAuth.

Для рівня абстракції 1 і 2 пропонуються різні версії SMADL. Для рівня абстракції 1 реалізовано візуальну мову (скорочено Visual SMADL або v-SMADL). Для рівня абстракції 2 реалізована текстова мова (текстовий SMADL або t-SMADL, для стислості).

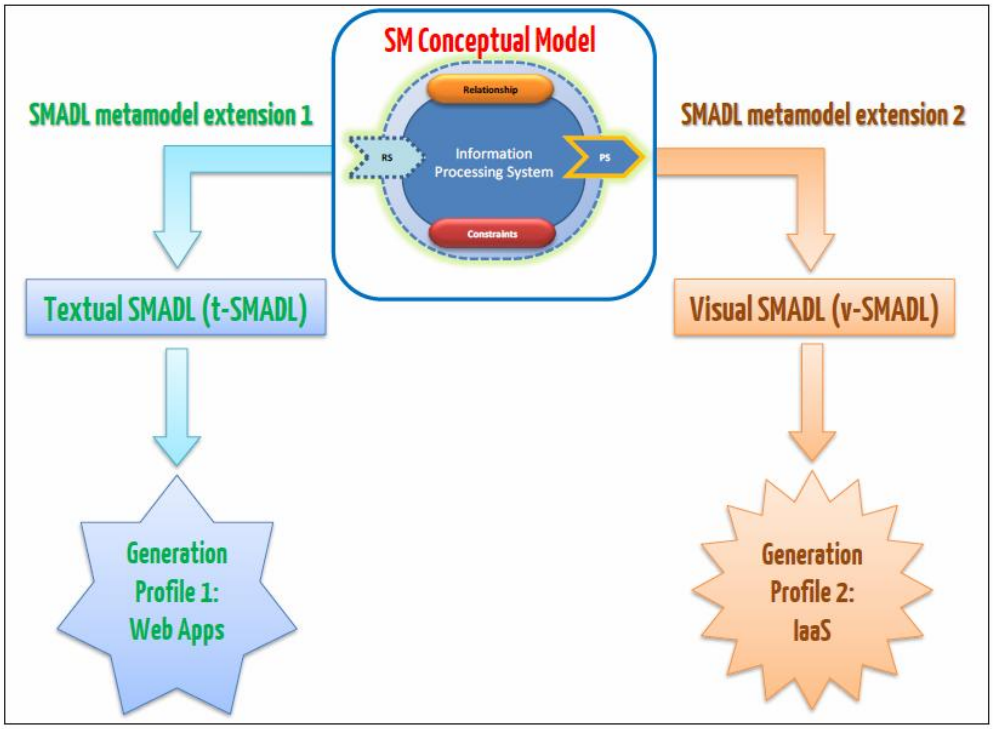

Рис. 2.7. Концептуальна модель SMADL, її версії, створені за допомогою розширення метамоделі, та їхні відповідні профілі генерації

SMADL був навмисно задуманий на рівні абстракції, щоб уможливити генерацію коду для різних контекстів, розширюючи його використання в майбутньому. Такі контексти охоплюються тим, що ми тут визначаємо як профіль покоління. У цій роботі профіль генерації — це набір перетворень, націлених на певний контекст, тобто кожна версія SMADL має свій власний

профіль покоління. t-SMADL використовує профіль генерації «веб-додатків», тоді як v-SMADL використовує профіль генерації «інфраструктура як послуга». Профіль генерації еквівалентний рівню абстракції 3. На рисунку 2.7 узагальнено асоціації різних версій SMADL та їхніх відповідних профілів.

Ці два профілі поколінь були визначені навмисно, значно відрізняючись один від одного. Створення профілів двох поколінь допомагає створити SMADL як практичне рішення та відповідний внесок у цю роботу. Далі наведено додаткові відомості про кожен профіль разом із деталями кожної версії SMADL.

Щоб зробити можливим створення різних версій SMADL, ми спочатку визначили загальну метамодель SMADL, яка показана на рисунку 2.8. Він розширений обома версіями SMADL і збирає всі елементи, представлені в концептуальній моделі SMADL, деталізуючи кожне взаємозв'язок між ними.

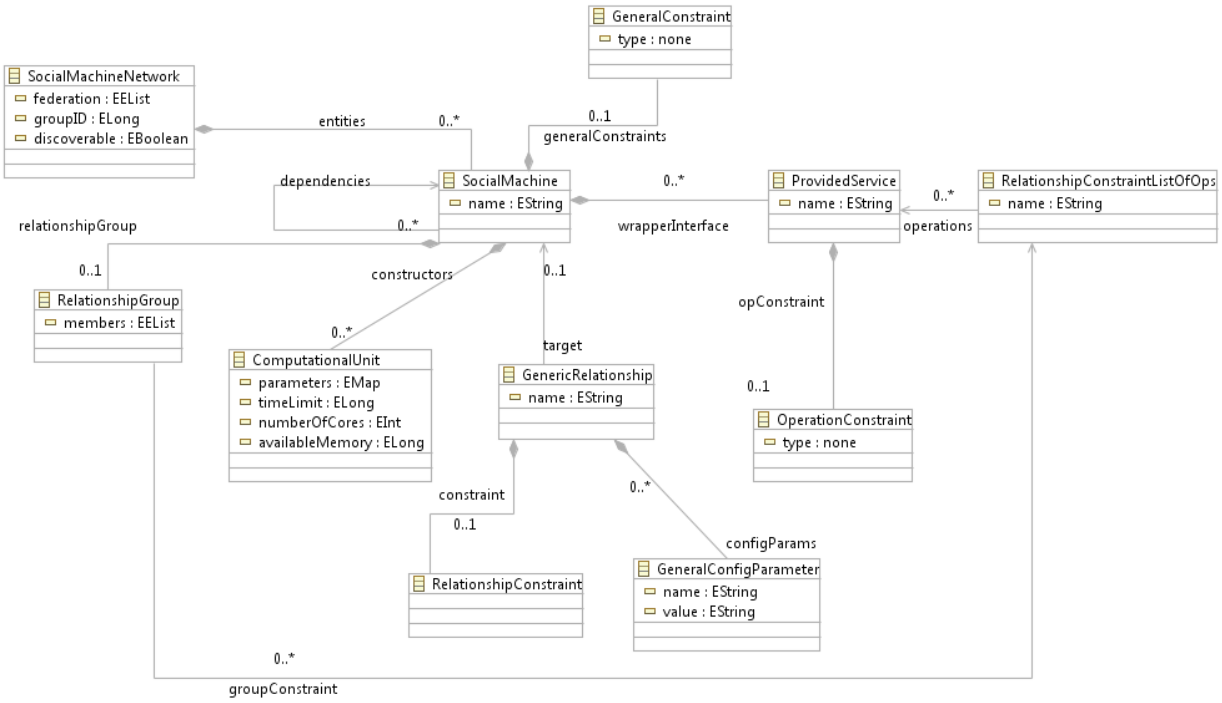

Рис. 2.8. Метамодель SMADL

Відповідно до загальної метамоделі SMADL, кожна соціальна машина має своє ім'я (не обов'язково унікальне) і може представляти деякі залежності для інших SM. Кожен SM може бути частиною мережі SM, яку можна

ідентифікувати за ідентифікатором групи та логічним атрибутом, який інформує, чи ця конкретна мережа доступна чи ні. SM може не мати жодного або багато конструкторів, які містять загальний список параметрів і три конкретні атрибути: часовий ліміт, кількість ядер і доступна пам’ять, що представляють загальні характеристики кожної соціальної машини, яку можна налаштувати в їхніх конструкторах. SM зазвичай надає набір послуг через свій інтерфейс оболонки. Кожна послуга є специфікацією операції, яка піддається деяким обмеженням. SM також може бути чутливим до загального обмеження, яке накладається на будь-яку надану послугу, яка не визначає її конкретне обмеження.

SM може бути частиною відносин, вимагати або надавати послуги. Також можливо, що будь-яка кількість соціальних машин має однакові відносини, створюючи групу відносин. Обмеження застосовуються до конкретних відносин або груп відносин.

Підводячи підсумок, рис. 2.8 показує загальні способи існування соціальних машин. Існує кілька контекстів (або доменів), у яких застосовна ця метамодель. Наступні описи двох різних версій SMADL є демонстрацією того, що можна розширити метамодель SMADL, щоб охопити специфічну поведінку будь-якого домену.

2.4 Реалізація метамоделі системи

t-SMADL представляє простий текстовий синтаксис на користь виразності. Мова може використовуватися як з низькорівневими конструкціями (умовними, так і без циклів). Концепції моделі SM безпосередньо відображаються в мові, полегшуючи новачкам використовувати її. У t-SMADL зв'язок представлений одним ключовим словом, тому можливості композиції для кількох SM можуть бути нескінченними. Подробиці представлені далі.

Для розробки t-SMADL було розглянуто ряд інструментів. Ґрунтуючись на наших результатах систематичного дослідження картографування, ми вирішили

використати мовний інструментарій Xtext, по-перше, тому, що він має велике визнання розробників, створюючи спільноту для співпраці навколо набору інструментів, і по-друге, через його актуальність, оскільки він був показаний на підручниках або міні-курсах на основних відповідних конференціях, таких як SPLC, OOPSLA, GPCE та EclipseCon.

Перший крок, який потрібно зробити при створенні нової мови, це визначення її граматики. Оскільки Xtext базується на Eclipse Modeling Framework (EMF), можна створити початкове визначення граматики на основі метамоделі EMF. На рис. 2.9 показано метамодель t-SMADL з деякими елементами, успадкованими від загальної метамоделі SMADL, як основу для граматики t-SMADL. Успадковані елементи виділені на схемі світло-сірим кольором.

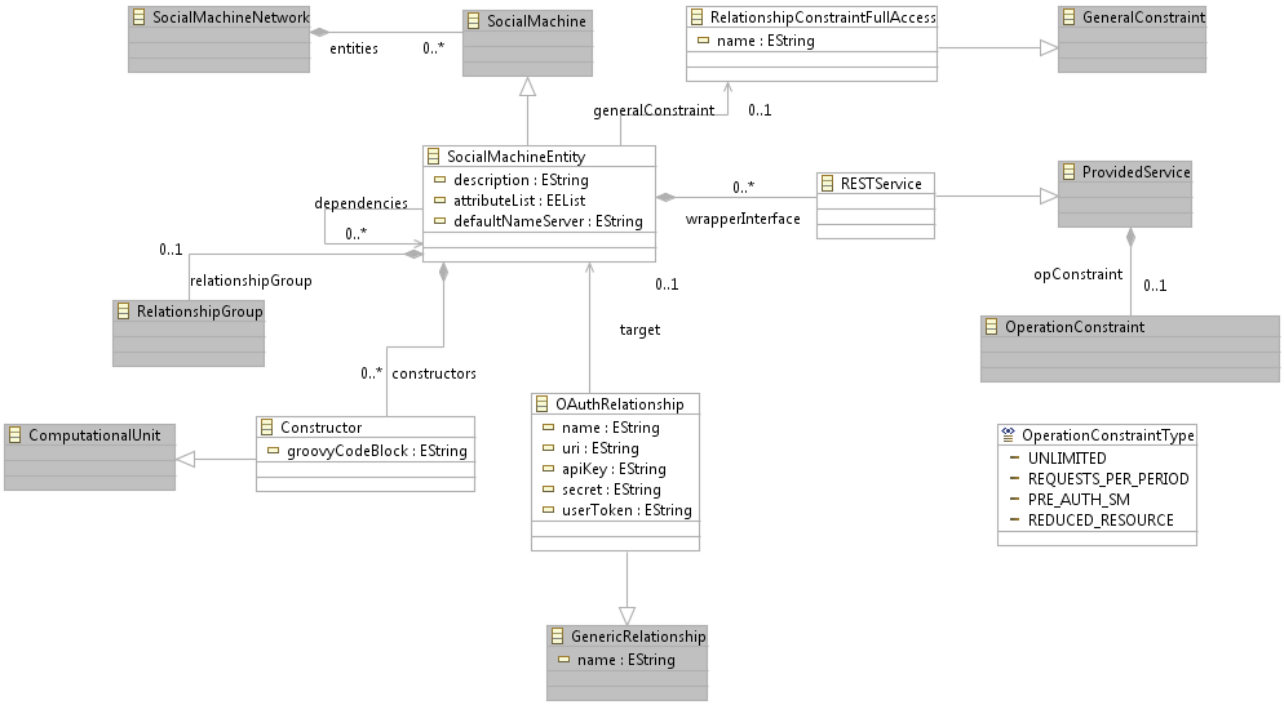

Рис. 2.9. Метамодель t-SMADL. Виділені структури (світло-сірим) успадковані від загальної метамоделі SMADL

У t-SMADL можна визначити мережу соціальних машин. Кожен SM встановлює зв’язок з іншими, як механізм імпорту в Java. Для початкової версії

t-SMADL ми визначили два типи зв’язків: загальний і на основі OAuth. Кожен встановлений зв'язок схильний мати власні обмеження. Обмеження можна застосувати до зв’язку, до всіх послуг, що надаються SM (загальне), і до конкретних послуг, що надаються (обмеження роботи). Загальний зв’язок і обмеження операції вже визначені в загальній метамоделі SMADL.

Кожна послуга, що надається, має власний підпис, і набір послуг, що надаються SM, становить її оболонковий інтерфейс. Для кращого розуміння метамоделі t-SMADL далі показано зразок коду, і кожна частина коду зіставлена з моделлю SM.

Відображення елементів соціальної машини на t-SMADL на прикладах

На рис. 2.10 показано фрагмент коду t-SMADL. Секції коду пронумеровані для полегшення подальших пояснень. t-SMADL не має формальної семантики, тому використані ключові слова було вибрано для представлення елементів соціальної машини зрозумілим і простим способом.

Сутність SM визначається за допомогою областей між фігурними дужками, дотримуючись синтаксису, подібного до Java. На рис. 2.10 сутність SM визначена та називається MyNewSocialMachine. Наступні елементи моделі SM відображаються на структурах t-SMADL:

• Система обробки інформації (IPS) → Частина 1 у фрагменті коду. t-SMADL дозволяє користувачеві визначити конструктор як обчислювальну одиницю для соціальної машини. Визначення конструктора необов’язкове, але, якщо це зроблено, він повинен містити тіло. Тіло сутності SM у t-SMADL написано мовою динамічної типізації на основі JVM під назвою Xbase, яка надається як частина фреймворку Xtext.

Код Xbase можна написати так само, як код Java. Важливо мати на увазі, що ми описуємо мову опису архітектури високого рівня, і такі низькорівневі конструкції, близькі до мови програмування загального призначення, не є частиною основної мети в цій роботі.

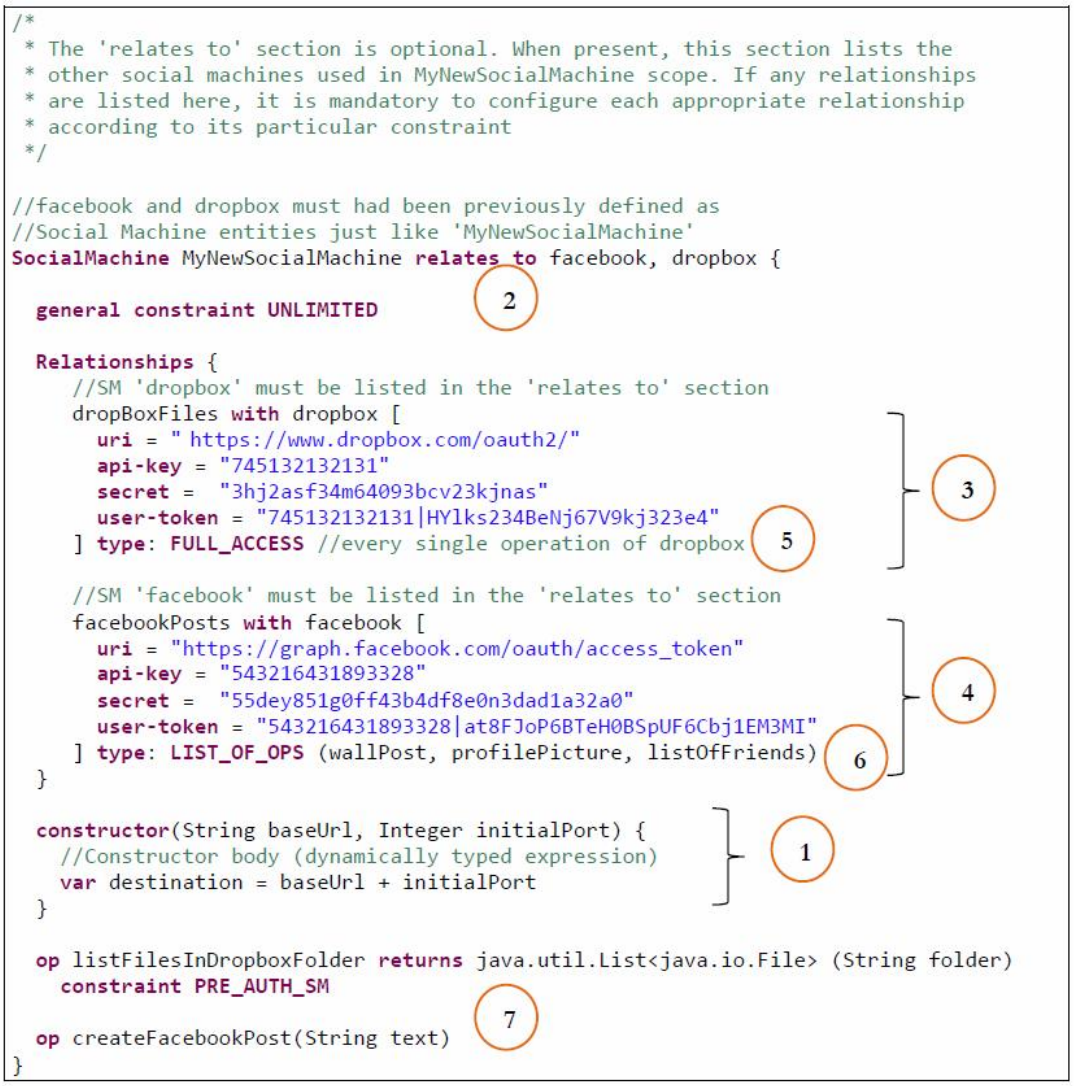

Рис. 2.10. Фрагмент коду t-SMADL

• Відносини (Rel) → Частини 2, 3 і 4 у фрагменті коду. Кожна організація SM повинна задекларувати інших SM, з якими вона підтримує стосунки, прямо за своїм визначенням (« стосується » у частині 2) до використання послуг, що надаються іншими. Також зауважте, що в цьому фрагменті коду facebook і dropbox повинні були раніше визначені як об’єкти соціальної машини, як і «MyNewSocialMachine». Частини 3 і 4 у фрагменті коду показують фактичну конфігурацію кожного зв’язку, який раніше називався переглядом взаємодії. t-SMADL визначає розділ коду « Відносини » для групування всіх переглядів взаємодії. Поточна версія t-SMADL дозволяє створювати два типи зв’язків: OAuthRelationship і GenericRelationship . Перший описує всі деталі, пов’язані з протоколом авторизації OAuth, який зазвичай використовується в поточних веб-

додатках. Останнє дозволяє встановлювати загальні зв’язки з будь-якою заданою кількістю параметрів. Єдина умова, щоб параметри не повторювалися себе. Використовуючи загальне відношення, можна використовувати t-SMADL для інших контекстів, крім програмованого Web, що робить мову розширюваною.

- Інтерфейс оболонки (WI) → Частина 7 у фрагменті коду. Набір наданих послуг складає інтерфейс оболонки разом із відповідними обмеженнями. Елемент WI моделі SM також використовується під час опису зв’язків (OAuth або Generic), оскільки вони абстрагують основні проблеми зв’язку.
- Надані послуги (PS) -> Частина 7 у фрагменті коду. У t-SMADL надані послуги абстрактно визначені з точки зору їх підпису та будь-яких обмежень на нього. Зауважте, що при визначенні PS немає необхідності встановлювати зв’язки. Відповідальність за фактичне оголошення стосунків несе запитувач послуг. Щоб визначити відкриті загальні служби в SMADL, користувачеві потрібно лише написати операцію без обмежень, як показано, наприклад, в операції createFacebookPost . Передбачається, що послуги, керовані зв’язками, визначаються за певними обмеженнями в провайдері та оголошуються в LIST_OF_OPS (частина 6) для кожного перегляду взаємодії в коді запитувача. Додаткові відомості можна знайти під час відображення елемента SM Constraints (C) на SMADL.
- Необхідні послуги (RS) → Частини 2 і 6 у фрагменті коду (неявно). Оскільки RS є необов’язковим елементом моделі SM, таким чином у t-SMADL набір необхідних послуг даного SM можна розглядати як частину його відносин. Ключове слово « стосується » (частина 2) містить список усіх SM, які використовуються в поточному SM. Потім, коли створюється подання взаємодії, наприклад « facebookPosts with facebook », ключове слово « with » обмежує, які операції, імовірно, будуть використані в обмеженні LIST_OF_OPS (частина 6), визначеному пізніше. У фрагменті коду перегляд взаємодії « facebookPosts » може отримати доступ лише до таких операцій « (wallPost, profilePicture,

listOfFriends) » із соціальної машини « facebook ». Це неявно визначає набір необхідних послуг у t-SMADL.

- Обмеження (C) → Частини 5 і 6 у фрагменті коду та інші частини, де зустрічається ключове слово " обмеження ". У t-SMADL обмеження можуть бути трьох типів: (1) загальне обмеження, застосовне до всіх наданих послуг; (2) обмеження операції, застосовне до однієї операції (наданої послуги) відразу, воно має вищий пріоритет, ніж загальне обмеження, тобто, коли операція оголошує обмеження, загальне обмеження більше не розглядається; і (3) обмеження зв'язку, яке обмежує, до яких операцій SM провайдера можна отримати доступ у даному поданні взаємодії. Для типів (1) і (2) t-SMADL наразі визначає чотири можливості: UNLIMITED (для необмеженого доступу), PRE_AUTH_SM (лише для попередньо авторизованих SM), REQUESTS_PER_PERIOD (обмежує кількість запитів до цієї операції за певний проміжок часу) і REDUCED_RESOURCE (обмежує ресурси, надані цією операцією, наприклад, зменшуючи пропускну здатність або потребуючи більше часу для повернення). Для типу (3) t-SMADL відкриває ще два ключові слова: FULL_ACCESS (без обмежень) і LIST_OF_OPS (за якими слідує список операцій, до яких має мати доступ цей зв'язок). Згідно з попередньо показаною моделлю SM, типи (1) і (2) обмежень є обмеженнями якості , а тип (3) є обмеженням видимості. Поточна версія t-SMADL не містить обмежень на основі функцій.

Ця характеристика в t-SMADL досягається через співвідношення SM. Композиція сутностей у t-SMADL подібна до композиції об'єктів в об'єктно-орієнтованій платформі. Об'єкти здійснюють виклики інших методів, а соціальні машини можуть здійснювати виклики наданих один одним сервісів через свій інтерфейс оболонки. Можна описати об'єднання SM за допомогою t-SMADL. Також можливо, що один SM об'єднує послуги іншого SM, таким чином надаючи нові послуги з можливою іншою функціональністю в іншому контексті. Це означає, що можна створювати нові SM на основі інших, застосовуючи добре відому в інформатиці концепцію «розділяй і володарюй».

t-SMADL — це мова на основі JVM. Це означає, що його можна повністю інтегрувати в Eclipse IDE, використовуючи всі переваги, які надає IDE, наприклад інтегрований редактор, налагоджувач та інші засоби. Крім того, мови на основі JVM скомпільовані в загальне середовище виконання (байт-коди), що полегшує інтеграцію такого типу мови. Це також відбувається з платформою Microsoft .NET і її спільною мовою виконання (CLR). Кожна мова на основі .NET компілюється до CLR, тому програмне забезпечення, написане за допомогою Microsoft Visual Studio .NET, може мати частини, написані різними мовами, оскільки ці мови компілюються безпосередньо до CLR. У нашому контексті можна інтегрувати t-SMADL з інша мова на основі JVM, як-от Groovy, Scala та сама Java, серед інших. Це надає SMADL невелику, але потужну неоднорідність.

Для об'єктно-орієнтованих розробників розуміння синтаксису t-SMADL не є проблемою. Оскільки мова має повну підтримку Eclipse і синтаксис, схожий на Java, вона може представляти плавну криву навчання. По суті, t-SMADL генерує код Java і, враховуючи, що мова сама по собі розроблена для опису архітектури програмного забезпечення, а не всієї системи, необхідно буде інтегрувати згенерований SMADL код із рукописним кодом. Таким чином, у цьому сенсі очікується, що розробники ООП ознайомляться з кодом t-SMADL.

Очікується, що код t-SMADL розвиватиметься разом із похідними програмами. Ця характеристика стає зрозумілішою, якщо зрозуміти, як t-SMADL використовується для генерації коду. Очікується, що саме визначення мови буде розвиватися, і це може зайняти багато часу для будь-якої мови. Однак використання мовного робочого середовища для розвитку мови може допомогти зменшити цей ризик (ERDWEG; FEHRENBACH; OSTERMANN, 2014). Оскільки t-SMADL підтримується мовним робочим середовищем, яке сприяє зворотній сумісності та створенню лексичних аналізаторів, аналізаторів і редакторів, розвиток мови не буде проблемою.

Будучи мовою на основі JVM, t-SMADL допомагає бути динамічною мовою. Спочатку t-SMADL допускає існування обчислювальної одиниці,

написаної на динамічно типізованій мові. І по-друге, найважливіше, t-SMADL можна поєднувати з методами впровадження залежностей, Google Guice 1 (вимовляється як «сік») або Spring Framework 1 2 .

За останнє десятиліття продуктивність JVM значно підскочила вперед і виявилася хорошим вибором для вирішення проблем масштабованості. Різні тести порівнювали мови на основі JVM, включаючи саму Java, з іншими мовами, такими як C, C++ і C#, а мови на основі JVM іноді навіть швидше, ніж інші мови, добре відомі своєю чудовою продуктивністю, головним чином тому , що багатьох оптимізацій, які з часом покращили продуктивність JVM, наприклад, Just-Компіляція часу, вдосконалені методи збирання сміття та динамічна повторна компіляція. У цьому контексті продуктивність є основною характеристикою для масштабованих мов. Деякі докази показують, що однією з найпопулярніших масштабованих мов є мова на основі JVM: Scala 1 . Репозиторій проектів GitHub 1 2 показує, що популярність Scala наближається до Java і C, а система пошуку роботи Indeed.com 3 показує, що кількість вакансій Scala потроїлася лише у 2012 році, досягнувши 100 000 розробників. Більше того, великий гравець Web 3.0 Twitter оголосив про те, що використовує Scala у своєму серверному програмному забезпеченні. t-SMADL — це текстова мова на основі JVM, і, як ми згадували в, її можна легко інтегрувати з іншою мовою на основі JVM, наприклад Scala. Обидва факти сприяють тому, що t-SMADL є мовою, сприйнятливою до збільшення складності та розміру.

Як уже було сказано, t-SMADL представляє заздалегідь визначений набір обмежень, які будуть використовуватися або для відносин соціальних машин, або для наданих послуг. Початковий набір обмежень було визначено на основі загальнодоступних API, розроблених такими великими гравцями, як Google, Twitter, Facebook тощо. Цей набір обмежень у мові, ймовірно, зростатиме, головним чином тому, що мова може почати використовуватися в різних контекстах, які ймовірно, призведе до нових концепцій і потребує абстрагування на рівні архітектури.

t-SMADL не є формальною мовою, визначеною в термінах алгебр процесів, таких як інші ADL, такі як Darwin, Wright або π -ADL. Ми відмовилися від формалізму на користь виразності та зрозумілості. Враховуючи складність і розмір веб-інформаційних систем, було б на порядок важче визначити такий формальний ADL для реалізації концепцій нашої моделі соціальної машини. Не кажучи вже про те, що крива вивчення мови буде непрактичною для виконання тематичних досліджень у часові рамки PhD.

2.5 Дослідження протоколів передача репрезентативного стану статичних та динамічних даних

OAuth 2.0 — це відкритий протокол авторизації, який дозволяє програмам отримувати доступ до даних один одного. Наприклад, ігрова програма може отримати доступ до даних користувача у Facebook або а програма на основі розташування може отримати доступ до домашнього та робочого місць користувача з Карт Google. На рис. 2.11 показано основну взаємодію між програмами, які використовують OAuth.

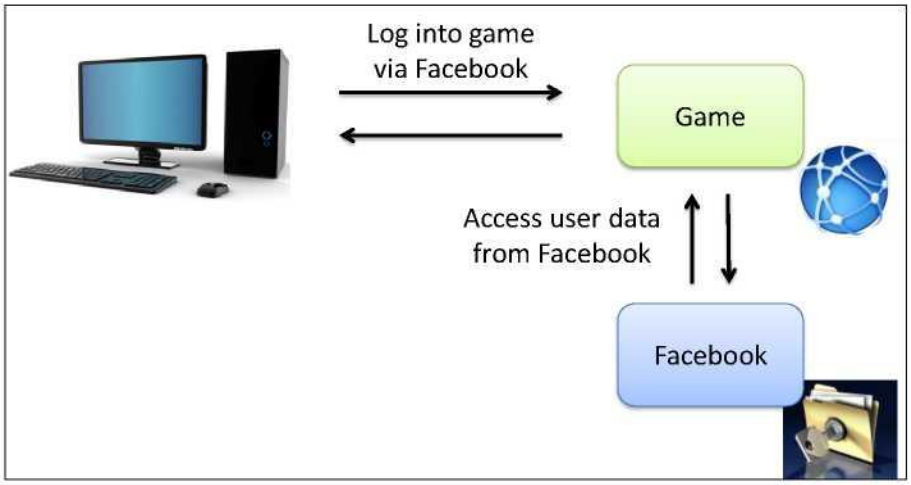

Рис. 2.11. Приклад використання OAuth для обміну інформацією

SMADL реалізовано за допомогою відносно простої граматики, яка сама по собі не може представляти повний потік інформації OAuth. Однак, окрім

мови та механізмів її перетворення, ми використовуємо допоміжні бібліотеки, які допомагають нам налаштувати постачальників OAuth із невеликою кількістю параметрів.

REST — це архітектурний стиль, розроблений як абстрактна модель веб-архітектури, щоб керувати нашим перепроектуванням і визначенням протоколу передачі гіпертексту (HTTP) і уніфікованих ідентифікаторів ресурсів (URI).

Принципи REST включають:

1) Концептуальні сутності та функціональні можливості моделюються як ресурси, визначені URI.

2) Доступ до ресурсів і керування ними здійснюється за допомогою стандартизованих добре відомих операцій HTTP (GET, POST, PUT і DELETE).

3) Компоненти системи спілкуються через ці стандартні операції інтерфейсу та обмінюються представленнями цих ресурсів (один ресурс може мати декілька представлень). У системі REST сервери та клієнти зазвичай переміщуються через різні стани представлень ресурсів, дотримуючись взаємозв'язків між ресурсами.

Завдяки цим простим, але потужним принципам REST досяг великої популярності головним чином тому, що йому не потрібен інший протокол зв'язку, натомість він використовує базові операції HTTP, повністю сумісні з усіма веб-браузерами.

t-SMADL застосовує принципи REST для створення служб RESTful (RICHARDSON; RUBY, 2007) з визначень SM. За замовчуванням надані послуги в а соціальна машина трансформується в RESTful-сервіси, які розробник повністю впровадить.

Xtext 1 — це фреймворк, який іноді називають мовним робочим середовищем, який значно скорочує зусилля зі створення хороших інструментів для мови. З граматики Xtext може генерувати парсер, серіалізатор і розумний редактор на платформі Eclipse. Усі проблеми самого Xtext і коду, створеного Xtext, можна налаштувати за допомогою ін'єкції залежностей. Для більшості проблем типова поведінка Xtext зазвичай є нормальною. Для проблем, які

потребують налаштування (перевірка, зв’язування/область тощо), Xtext надає простий у використанні API. Ще одна перевага фреймворку Xtext — це Xtend, динамічна мова на основі JVM, яка представляє потужні конструкції для генерації коду, наприклад, розширені рядки.

t-SMADL реалізовано як плагін Eclipse із вбудованим редактором, який забезпечує живий зворотний зв’язок під час введення коду. Він вбудований код автозаповнення та функції допомоги коду. На рис. 2.12 показано функцію завершення коду плагіна t-SMADL, інтегрованого в Eclipse IDE. Середовище вже виявляє соціальні машини, визначені в тому самому проекті, і дозволяє розробнику посилатися на них у розділі « стосується ». На рис. завершення коду показує параметри: Dropbox , GoogleMaps , SMMashup і facebook , які є об’єктами SM, визначеними в одному проекті, на які можна посилатися.

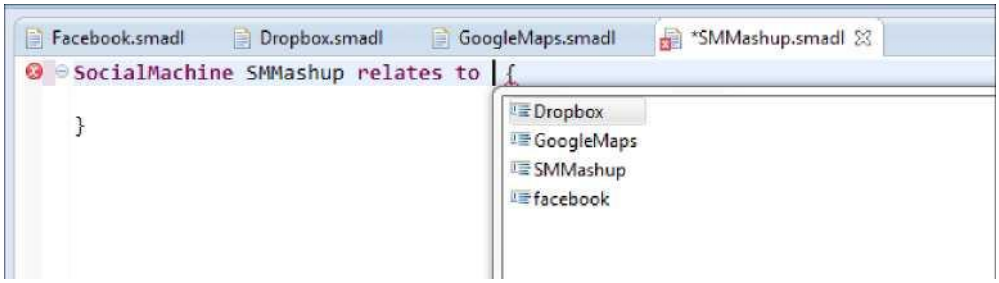

Рис. 2.12. Функція завершення коду t-SMADL

Інструмент забезпечує перевірку синтаксису, а також деяку семантичну перевірку з послідовними повідомленнями про помилки. На рис. 2.13 показано синтаксичну та семантичну помилку під час визначення та зв’язку OAuth. На рис. 2.13 соціальна машина під назвою SMMashup підтримує зв’язки з двома іншими SM: Dropbox і Facebook . Потім у розділі «Зв’язки» створюється подання взаємодії під назвою dropBoxFiles , яке встановлює зв’язок із Dropbox. Під час визначення цього перегляду взаємодії можна обмежити операції, до яких передбачається отримати доступ у Dropbox, за допомогою обмеження LIST_OF_OPS . Проблема виникає через відсутність операції dropBoxFiles,

визначеної в Dropbox SM (синтаксична помилка), і лише операції цільового Dropbox дійсні для цього зв’язку (семантична помилка).

Рис. 2.13. Синтаксична та семантична помилка, перевірена в плагіні t-SMADL

v-SMADL — це візуальна мова для керування інфраструктурами центрів обробки даних, що дозволяє автоматично розгортати повну інфраструктуру лише за допомогою графічного представлення. Концептуальну модель SM можна безпосередньо відобразити в мові та її візуальних елементах, як ми вже показали з t-SMADL. У v-SMADL зв’язок представлений взаємозв’язками між віртуальними машинами та групами віртуальних машин під назвою VM Pools.

Для розробки v-SMADL ми використовували Eclipse Modeling Framework (EMF) у поєднанні з Eclipse Graphical Modeling Framework (GMF), яка надає набір генеративних компонентів та інфраструктури виконання для розробки графічних редакторів на основі EMF і Graphical Editing Framework (GEF).).

Під час використання GMF першим артефактом, який необхідно розробити, є метамодель EMF (файл .ecore), на відміну від автоматично згенерованої метамоделі t-SMADL, яка походить від граматики Xtext. З моделі «.ecore» походять і налаштовуються всі інші моделі, наприклад панель інструментів (файл .gmftool) і графічні елементи (файл .gmfgraph). малюнок 2.10 показано метамодель v-SMADL ecore як розширення загальної метамоделі SMADL. Зауважте, що наступні сутності: об’єкт VM у v-SMADL є розширенням

об’єкта SM; об’єкт Template, застосовний для кожної віртуальної машини, насправді є обмеженням загальної метамоделі SM; об’єкт Datacenter – це розширення мережі CM; і об'єкт мережевого інтерфейсу насправді є розширенням групи зв'язків, яка пояснювалася раніше. Крім того, кожна сутність метамоделі v-SMADL реалізує метод під назвою «validate». Ці методи використовуються для перевірки всієї діаграми v-SMADL перед її розгортанням на сервері.

У цій послідовності ми детально описуємо кожен елемент метамоделі v-SMADL, яка представляє елементи базової інфраструктури центру обробки даних, а потім представляємо концептуальне відображення цієї метамоделі в концептуальну модель SM.

Елементами інфраструктури центру обробки даних є:

- Віртуальна машина (VM):— це програмна емуляція комп’ютера. Віртуальні машини працюють на основі комп’ютерної архітектури та функцій реального чи гіпотетичного комп’ютера.
- Пул віртуальних машин (VM Pool): сукупність віртуальних машин зі спільними атрибутами, як-от одна мережа або однакові параметри брандмауера.
- Балансувальник: балансування навантаження — це метод комп’ютерної мережі для розподілу робочого навантаження між кількома обчислювальними ресурсами, такими як комп’ютери, комп’ютерний кластер, мережеві зв’язки, центральні процесори або диски. Балансування навантаження має на меті оптимізувати використання ресурсів, збільшити пропускну здатність, мінімізувати час відповіді та уникнути перевантаження будь-якого з ресурсів. Використання кількох компонентів із балансуванням навантаження замість одного компонента може підвищити надійність завдяки резервуванню. Балансувальник навантаження зазвичай реалізується спеціальним програмним або апаратним забезпеченням, таким як багаторівневий комутатор або серверний процес системи доменних імен.
- Мережа: комп’ютерна мережа або мережа даних — це телекомунікаційна мережа, яка дозволяє комп’ютерам обмінюватися даними. У

цьому контексті елемент «Мережа» представляє комп’ютерну мережу з ідентифікатором віртуальної локальної мережі (LAN), публічною та приватною адресами.

- Адреса: представляє IP-адресу в мережевому інтерфейсі. Мережевий інтерфейс — це системний (програмний та/або апаратний) інтерфейс між двома частинами обладнання або рівнями протоколу в мережі. IP-адреса — це числова мітка, призначена кожному пристрою (наприклад, комп’ютеру, принтеру), який бере участь у комп’ютерній мережі, яка використовує Інтернет-протокол для зв’язку.

Усі обмеження v-SMADL перевіряються на вимогу під час версії моделі. На відміну від t-SMADL, контекст, де використовується v-SMADL, не вносить великої різноманітності у визначення служби, яка, до речі, представлятиме різні типи обмежень.

Подібно до t-SMADL, v-SMADL не є формальною мовою, визначеною в термінах алгебр процесів, таких як інші ADL, такі як Darwin, Wright або π -ADL. Також у цьому випадку ми відмовилися від формалізму на користь виразності та зрозумілості. Дійсно, адміністрування центру обробки даних є динамічним завданням, яке не обов’язково потребує формально визначеної мови. Простого факту, що v-SMADL є візуальним, достатньо для залучення нових користувачів. У порівнянні з іншими великими гравцями віртуалізації серверів і корпоративних хмарних обчислень, такими як OpenNebula 1 , OpenStack 1 2 , Apache CloudStack 3 і Eucalypts 4 (Amazon AWS), візуальних інструментів для завдань адміністрування центру обробки даних мало.

v-SMADL використовує профіль генерації «Інфраструктура як послуга» (IaaS). У цьому сенсі SMADL використовується для визначення, як соціальних машин, внутрішніх елементів центру обробки даних та їхніх зв’язків, включаючи віртуальні машини, пули віртуальних машин, балансувальники навантаження, мережі та адреси.

Цей профіль призначений для інтерпретації діаграми v-SMADL і викликів попередньо визначеної служби REST, яка створює екземпляр XenServer 5 ,

забезпечуючи налаштування центру обробки даних відповідно до конфігурації, наданої користувачем для кожної віртуальної машини та її відповідників. Для цього профілю реалізовано графічний інтерфейс користувача, повністю інтегрований у платформу Eclipse.

Щоб зробити v-SMADL практичним, інструмент редактора діаграм був реалізований і названий uCloud Console. Майно програмного забезпечення uCloud належить компанії USTO.RE. Рис. 2.14 дає уявлення про архітектуру uCloud.

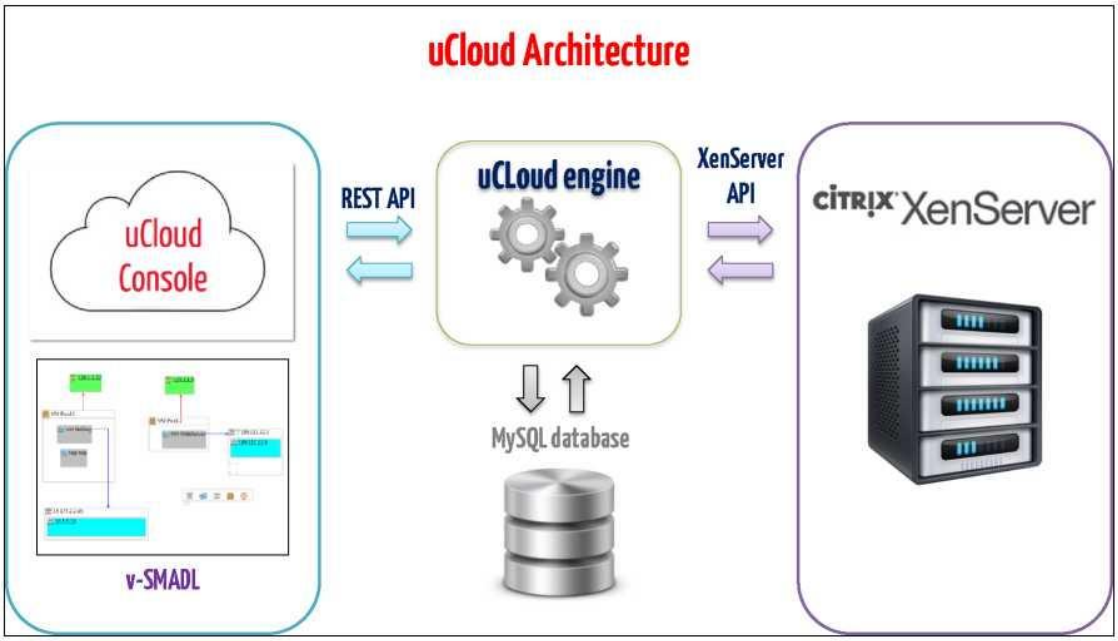

Рис. 2.14. Архітектура програмного забезпечення uCloud. v-SMADL реалізовано в модулі uCloud Console

uCloud Console представляє палітру інструментів із швидким доступом до всіх візуальних елементів діаграми v-SMADL. Крім того, якщо покажчик миші зупиняється на деякий час у порожніх частинах діаграми, користувачеві пропонується ярлик панелі інструментів. На рис. 2.15 показано ярлик панелі інструментів (ліворуч) і палітру (праворуч). Палітра також містить функцію додавання приміток користувача до діаграми (жовтий значок у верхньому правому куті).

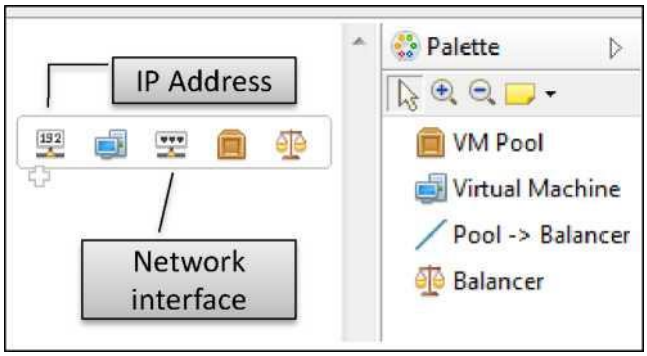

Рис. 2.15. Ярлик панелі інструментів uCloud Console і палітра інструментів

Щоб уявити, як працює консоль uCloud, на рис. 2.16 представлено її інтерфейс. Зліва діаграму v-SMADL можна редагувати. У центрі палітра інструментів дозволяє користувачеві додавати нові візуальні елементи (пул віртуальних машин, віртуальну машину, мережу та IP-адресу) і з'єднання між віртуальними машинами та мережами, а також між пулами віртуальних машин і балансувальниками. Важливо зауважити, що зв'язок між пулами віртуальних машин і віртуальними машинами, а також між мережами та IP-адресами здійснюється за допомогою обмеження, тобто користувачеві потрібно лише додати вмістомий елемент у область контейнера.

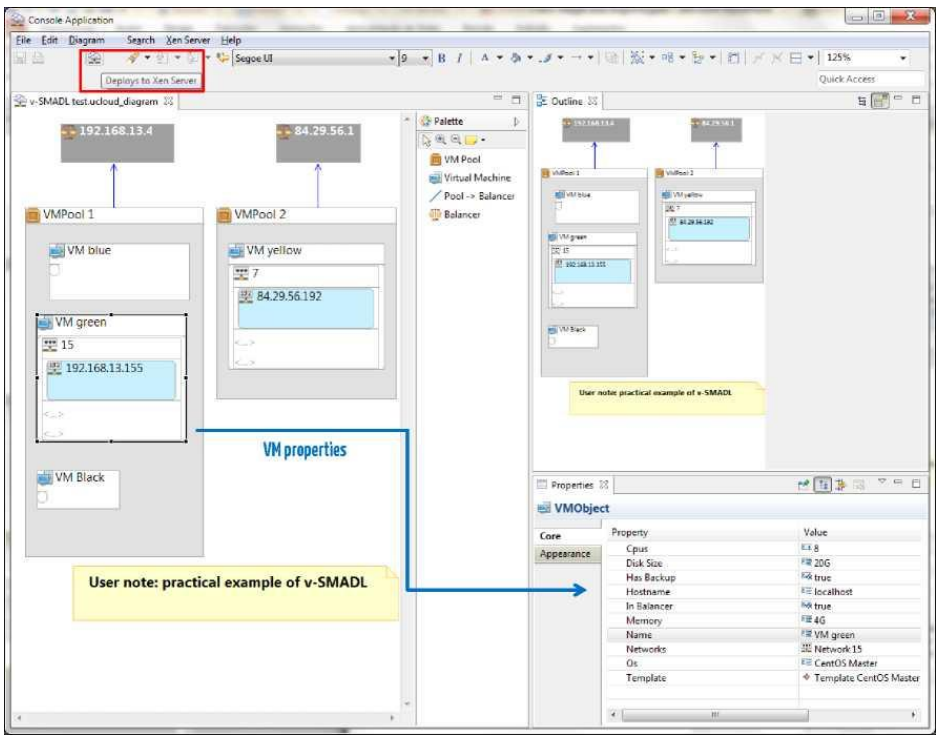

Рис. 2.16. Інтерфейс uCloud Console

І останнє, але не менш важливе: на рис. 2.16 у верхньому лівому куті виділено кнопку «Deploys to Xen Server», яка викликає REST API, доступний за допомогою компонента механізму uCloud. У свою чергу механізм uCloud обробляє запит, зберігає діаграму користувача для подальшого використання та викликає власний API XenServer, щоб розпочати розгортання кожного пулу віртуальних машин, включаючи його віртуальні машини та балансувальник, а також конфігурацію пов'язаної мережі/адрес. Віртуальні машини автоматично запускаються на XenServer.

Ми визначили SMADL — мову опису архітектури соціальних машин — як спробу створити абсолютно інший спосіб програмування Інтернету, змішуючи концепції ADL і DSL. Як ADL, він дозволяє описувати соціальні машини (та їх мережі) у термінах зв'язків як абстракції високого рівня, без необхідності вказувати деталі зв'язку (протоколів) та/або методів автентифікації. Як DSL, він дозволяє впроваджувати та інтегрувати веб-сервіси за допомогою динамічно введеного синтаксису, повністю інтегрованого у віртуальну машину Java та IDE Eclipse. Крім того, елементи моделі соціальної машини були відображені на конструкціях SMADL і представлені тут.

Висновки до розділу

В даному розділі наведені методи та засоби імплементації соціальної взаємодії отримання статичних та динамічних даних. Виконано дослідження основних характеристики соціальних машин, наведені типи та приклади соціальних машин як джерел отримання статичної і динамічної інформації. Виконано визначення мови для опису архітектури інформаційних систем на основі Web 3.0, досліджено протоколи передача репрезентативного стану статичних та динамічних даних.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МЕТОДОЛОГІЇ КОНТРОЛЮ ДОСТУПУ ТА ЗАХИСТУ СТАТИЧНОЇ І ДИНАМІЧНОЇ ІНФОРМАЦІЇ В СОЦІАЛЬНИХ МЕРЕЖАХ

3.1 Аналіз загроз інформації в соціальних мережах

Соціальна мережа – соціальна структура, утворена індивідами або організаціями в якій підтримуються соціальні відносини. Вона відображає різноманітні зв'язки між ними через різноманітні соціальні взаємовідносини, починаючи з випадкових знайомств і закінчуючи тісними родинними зв'язками. Вперше термін було запропоновано в 1954 році Дж. А. Барнесом (в роботі Class and Committees in a Norwegian Island Parish, «Human Relations»).

Зведена інформація компанії Microsoft щодо кількості та типу вразливостей в відповідних продуктах наведена на графіку рис. 3.1.

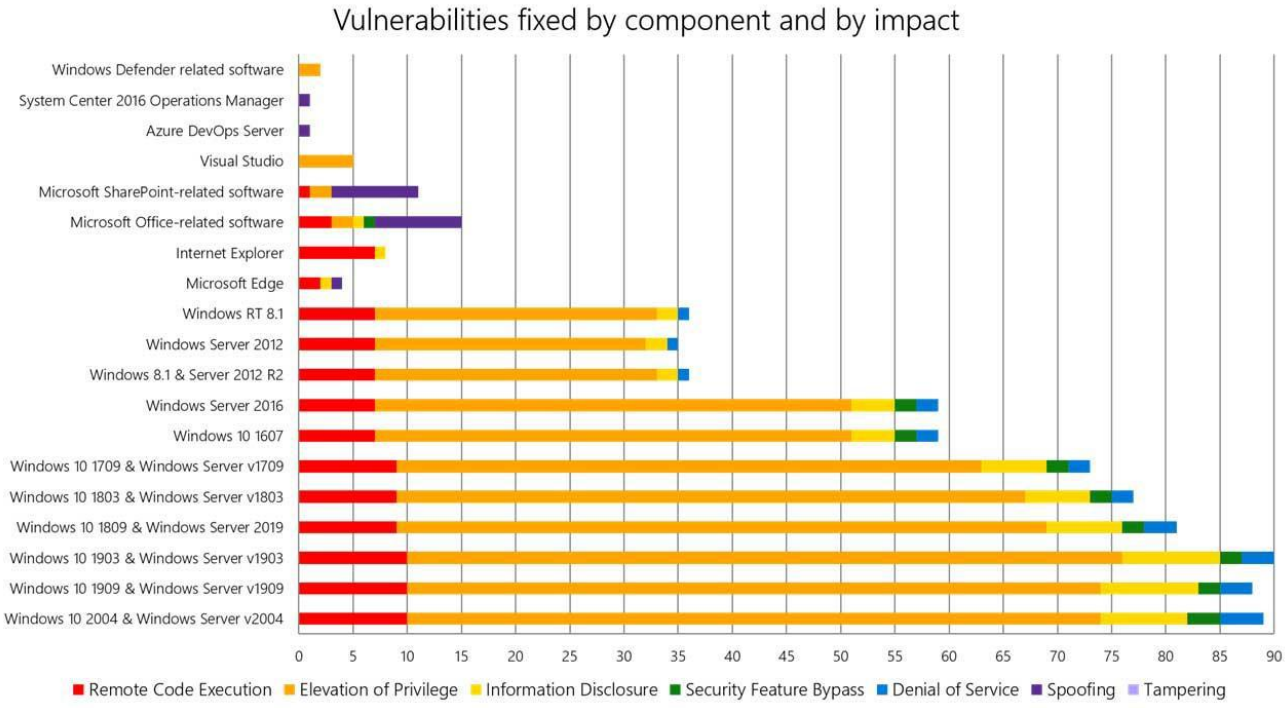

Рис. 3.1. Зведена інформація фірми Microsoft за вразливостями

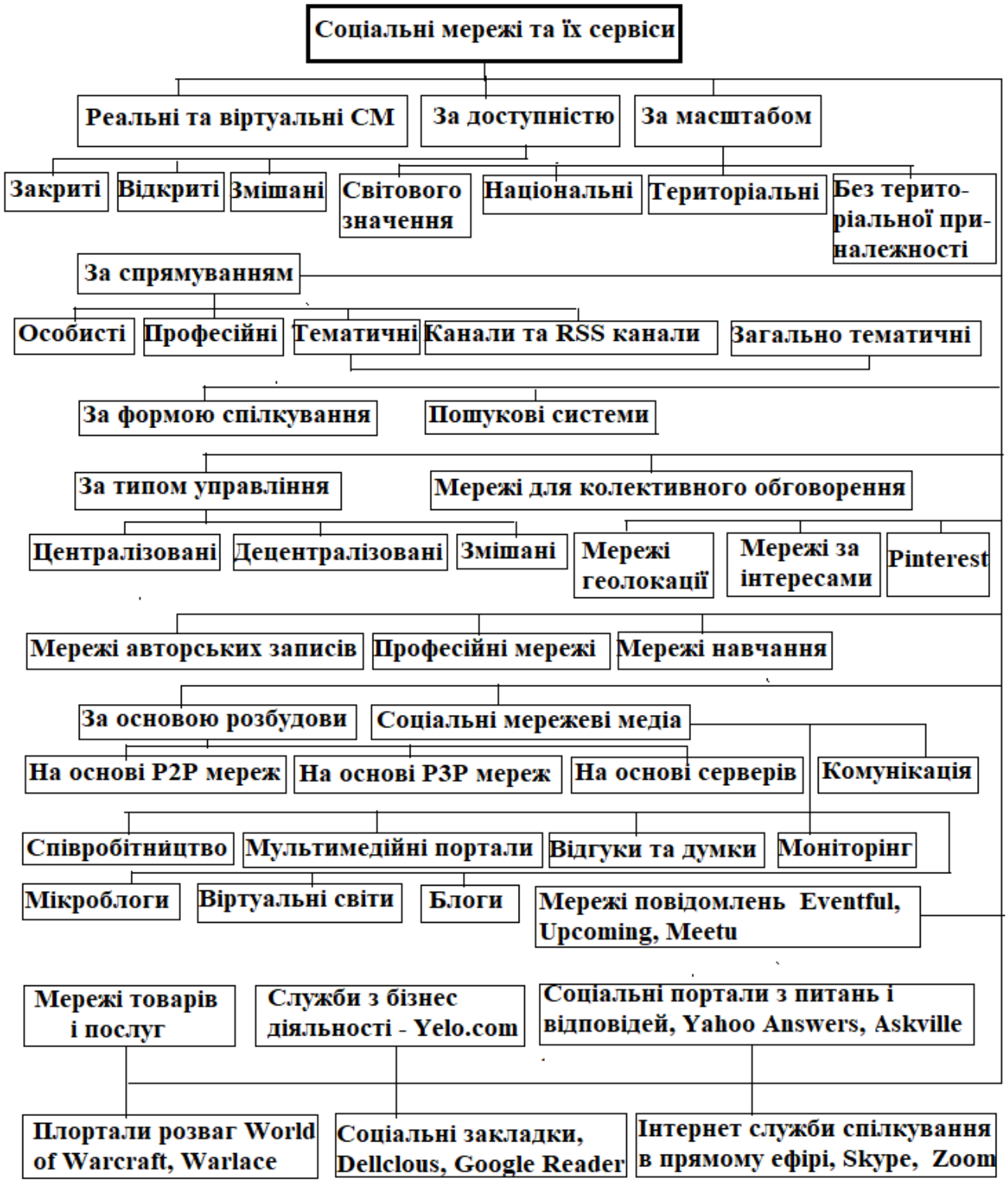

Рис. 3.2. Фрагмент класифікації соціальних мереж (СМ) та їх сервісів

Мотиви, цілі збору даних. Зростає частка впливів, спрямованих на крадіжку інформації (рис. 3.3).

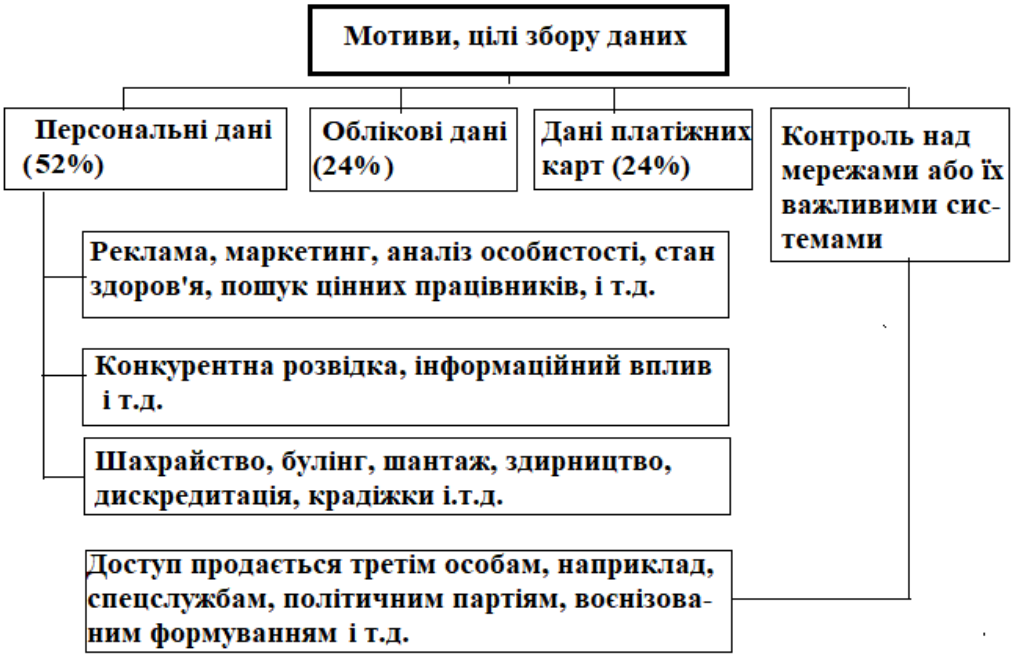

Рис. 3.3. Мотиви і цілі збору динамічних та статичних даних

Контроль над мережами та їх важливими системами, цей тип впливів з’явився в останні роки. Активні користувачі соцмереж мають на 30% вищий рівень ризику стати жертвами, тому що їх інформація частіше за інших може бути розкрита.

Наприклад, користувачі Facebook, Instagram і Snapchat мають на 46% вищий рівень ризику «захоплення» акаунта, ніж ті, у кого немає акаунтів в цих соціальних мережах. Інноваційний підхід Facebook в сфері збору інформації зробив соціальну мережу вкрай зручною платформою для бізнесу, зацікавленому в маркетингу, рекламі і просуванні своєї продукції. Так, користувачі Facebook добровільно надають інформацію про свої дні народження, місця проживання, сім'ю, працевлаштування і інтереси, підкріплюючи ці дані особистими фотографіями, повідомленнями і статусами.

Більш того, платформа також збирає і зберігає інформацію про переваги, повідомленнях і дзвінках, списки друзів, сім'ї і колег, пошукову історію, а також точно знає, які ЗМІ користувачі читають, які заклади відвідують найчастіше і з яких пристроїв заходять в соціальну мережу.

Друга загроза – це так званий маскарад або можливість підміни особистості: достеменно неясно, хто саме приховує свої дії під ім'ям друзів або прикривається фотографіями знайомих в профілі соцмережі. Якщо при листуванні по електронній пошті можна було б за IP–адресою відправника зібрати про нього хоча б якусь інформацію, то в соцмережі і цього не вийде.

Сценарій подібного маскараду можливий і на корпоративному рівні.

Результатом такого шкідливого сценарію може стати фішинг, організація «чорного піару» або «антипіару» [16]. Уже було чимало прикладів, коли незрозуміло ким створюється сайт від імені якоїсь компанії, – це породжує проблему для початкового бренду.

Третя загроза пов'язана зі зломом призначених для користувача записів соціальних ресурсів. За допомогою злому [2] зловмисник може проникнути в соцмережу (в тому числі від імені того, хто представляє в ній компанію, організацію або бренд), розіслати по її списку друзів фітингових повідомлень і отримати гроші або мотивувати одержувачів до яких–небудь негативних дій – зокрема, пройти за вказаним URL і запустити шкідливий код.

«Останнім часом особливою популярністю користуються сервіси для скорочення довжини URL, що дозволяють замаскувати адресу небажаного сайту під коротким посиланням. Сьогодні йде активна боротьба з цими ризиками – сервіси скорочення URL стали застосовувати поліпшені механізми детектування спаму і інших погроз. Однак для користувачів соціальних мереж загроза залишається – привабливі повідомлення і пропозиції від уже відомих вам контактів, які були зламані, часто призводять до завантаження зловмисних програм або демонстрації небажаних сторінок Інтернету».

Соціальні мережі являють собою квінтесенцію сучасних Web–технологій. Вони об'єднують в собі і всі загрози, властиві Інтернету. Їх можна розділити на наступні великі групи:

• Web–атаки. Оскільки соціальні мережі – це Web–додатки, то їх можуть використовувати хакери, щоб організувати атаки на уразливості в браузерах.

Інструментами для таких атак можуть бути троянські програми, фальшиві антивіруси, соціальні черви, які використовують для свого поширення списки друзів, та ін. Їх основна мета – проникнути в інформаційну систему відвідувача соціальної мережі і закріпитися в ній. Для захисту використовуються такі традиційні засоби, як антивіруси, які вміють працювати в режимі реального часу і блокують завантаження шкідливих кодів.

• Крадіжка паролів і фішинг. Оскільки для ідентифікації соціальні мережі використовують паролі, то досить дізнатися цю найзаповітнішу послідовність символів – і можна від чужого імені розсилати рекламу і робити інші (часто заборонені) справи. Крім того, деякі компанії використовують соціальні мережі для просування власної продукції, а кража пароля адміністратора групи дозволяє, по суті, вкрасти і саму групу. А для отримання персональної інформації традиційно використовують фішинг, підставні сайти, соціальну інженерію і багато іншого.

Проблема клонування. Під професійним клонуванням ми розуміємо особливий тип атаки уособлення, який відбувається в межах однієї платформи ICM, як зображено на рисунку 3.4. Метою супротивника тут є створення відвідувача для певного користувача, який вже має деяку дійсну програму в одній мережі. З технічної точки зору, ця атака може бути реалізована через реєстрацію нового продукту, використовуючи той самий (або подібний) вміст, що і існуючий.

Рис. 3.4. Атаки підробленого вступу: жертва U не має жодного облікового запису ICM, жертва V має обліковий запис на ICM1, а жертва Z – на ICM2.

3.2 Моделі захисту статичної і динамічної інформації у соціальних мережах

Методи та засоби захисту інформації в соціальних мережах представлені на рис. 3.5.

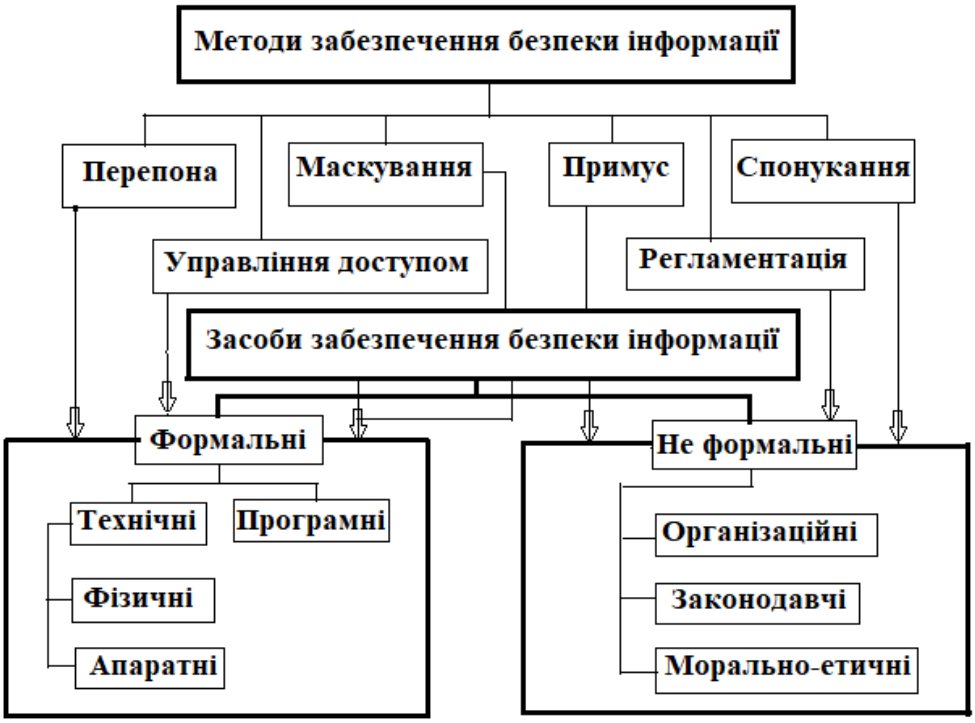

Рис. 3.5. Методи та засоби захисту інформації в соціальних мережах

Рішення інформаційної безпеки більше не можуть грунтуватися на одній технології: вони вимагають підходу, заснованого на багаторівневої технології, в поєднанні з позицією «нульової довіри» (zero trust), щоб зупинити можливі порушення безпеки. Ці багаторівневі технології забезпечують безпрецедентний рівень контролю, видимості і гнучкості. Це те, що необхідно в динамічній війні проти невідомих зловмисників.

Із аналізу літератури, захист мережі можна розділити на сім рівнів складності:

Перший – самий елементарний і обов'язковий. Тут головний інструмент захисту – firewall. Firewall повинен лімітувати використання сервісів, які надаються користувачам. Також firewall повинен стежити за всіма з'єднаннями, як з одного, так і з іншого боку.

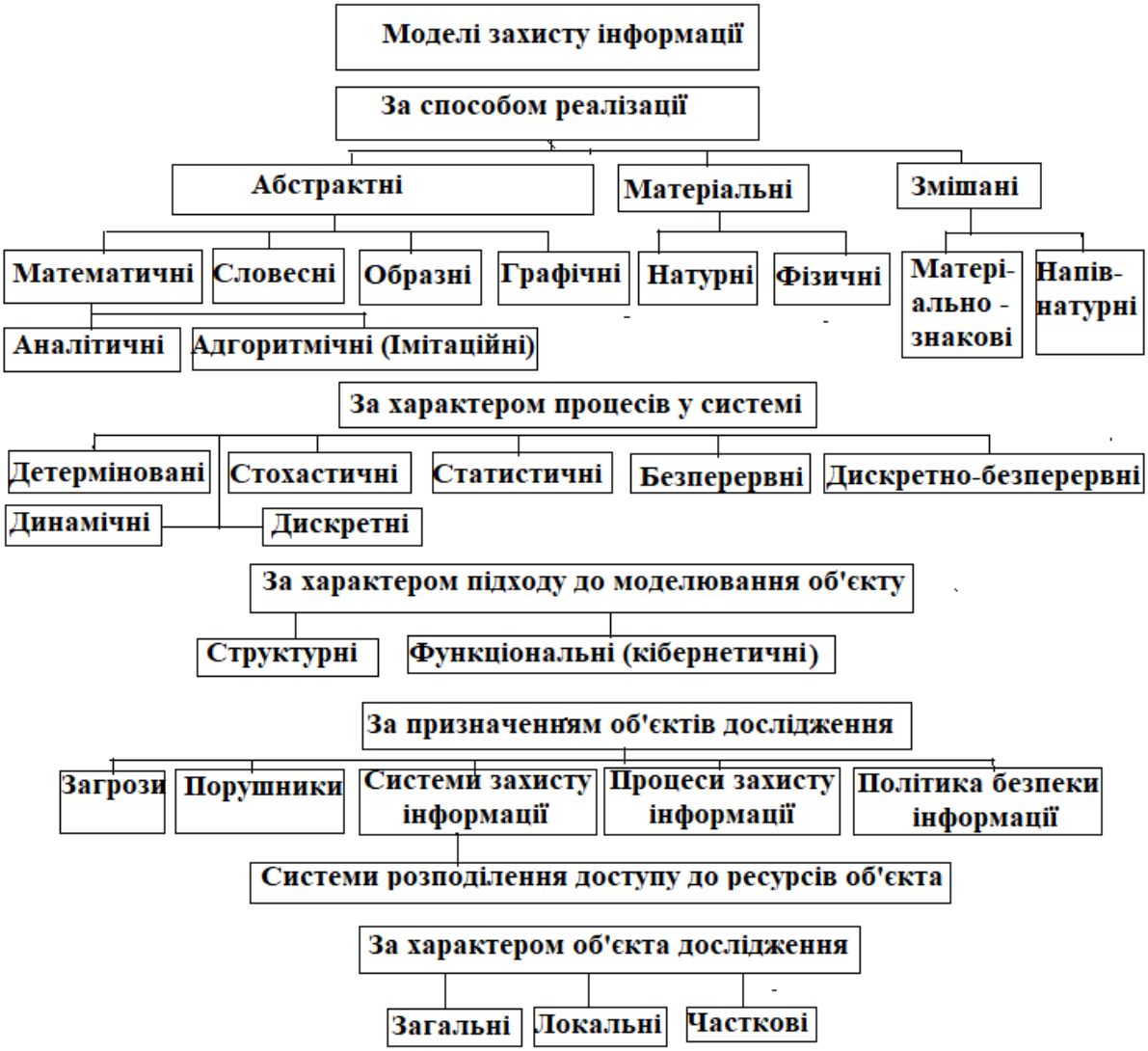

Рис. 3.6. Класифікація моделей захисту інформації

Другий рівень безпеки передбачає конфігурацію операційної системи, під чиїм керівництвом працює веб–сервер. Кожна операційна система дозволяє створювати контрольні листи безпеки (security checklist). Ці установки повинні бути узгоджені з операційними системами вендорів, які співпрацюють з компанією.

Третій рівень орієнтований вже на мережу. Необхідно оснастити датчиками атаки мережевого обладнання та програмного забезпечення провайдера, що забезпечує хостинг. Головне, щоб сигнал який надійшов про небезпеку був правильно оброблений і нейтралізована небезпека.

Четвертий рівень безпеки – установка програмного забезпечення на рівні хостингу. Це значно складніше завдання. По–перше, тут можна зіткнутися з запереченнями самої хостинг–компанії. А по–друге, таке програмне забезпечення набагато складніше простих датчиків. З цієї причини і рівень безпеки вищий.

Тому пропонується метод захисту інформації інформації на базі нового протоколу обміну даних. Він заснований на модифікації відомого алгоритму (OFM) S-box ГОСТ 34.12-2015, що забезпечує "усунення" можливих криптографічних закладок та підвищення криптостійкості в пост квантовий період (поява повномасштабного квантового комп'ютер, що дозволяє зламати на основі алгоритмів Гровера та Шора сучасні симетричні та асиметричні криптосистеми). Крім того, комерційне впровадження забезпечить "протидію" можливих криптодепозитів спецслужбами, що зменшить ризик злому шляхом виявлення "слабких" (вразливих) місць на основі криптографічних закладок.

В алгоритмі блок, що шифрується (довжина 64 біта), розділений на дві рівні частини (32 біти) - праву та ліву. Далі тридцять дві ітерації виконуються з використанням ітераційних ключів, отриманих з вихідного 256-бітного ключа шифрування. Під час кожної ітерації здійснюється одне перетворення на основі мережі Фейстела з правою та лівою половиною зашифрованого блоку. Спочатку права частина складається в модуль 232 з поточним ітераційним ключем, потім отримане 32-бітове число ділиться на вісім 4-бітових і кожен з них, використовуючи таблицю перестановок, перетворюється в інший 4-бітний номер. Після цього перетворення отримане число крутиться вліво на одинадцять розрядів. Далі XOR трансформується з лівою половиною блоку. Отримане 32-бітове число записується в правій половині блоку, а старий вміст правої

половини переноситься в ліву половину блоку. Діаграма основного кроку крипто-перетворення алгоритму показана на рис. 3.7.

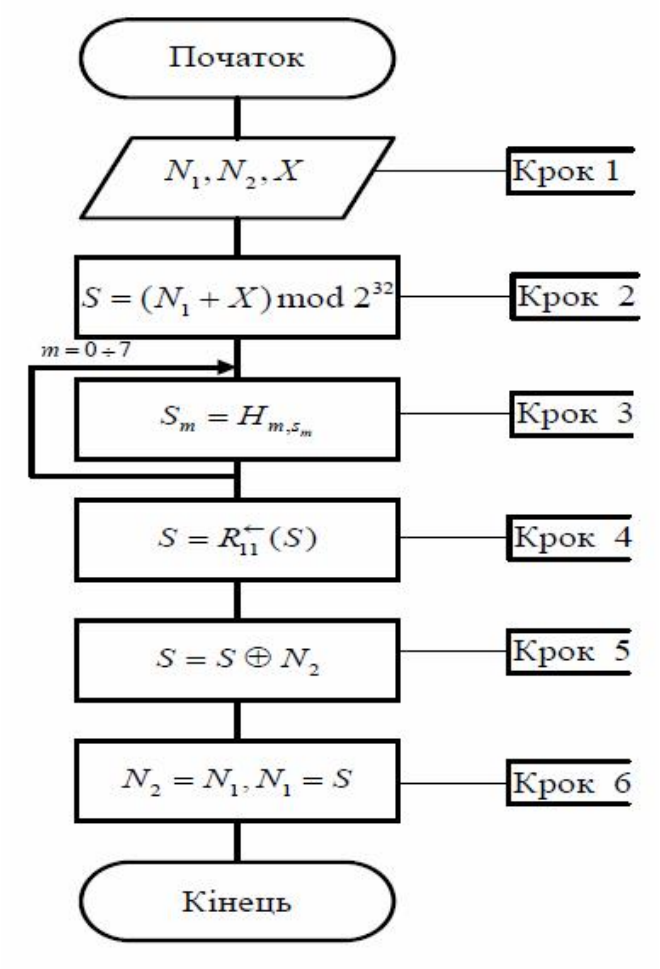

Рис. 3.7. Схема основного кроку криптоперетворення алгоритму

Основний крок криптотрансформації алгоритму складається з наступних етапів:

Крок1. Введення вихідних даних для основного кроку криптоперетворення - 64-розрядний блок введення перетворюється на два 32-розрядних цілих числа (молодшу і найстаршу частини);

Крок 2. Додавання до ключа. Молодша частина перетвореного блоку складається в модуль із ключовим елементом, що використовується на кроці.

Крок3. Заміна блоку. Отримане на попередньому кроці 32-бітове значення інтерпретується як масив із чотирьох 4-бітових блоків коду: Sm =(S0,S1,S2,...S15).

Крок4. Циклічний зсув на 11 біт вліво.

Крок5. Додане побиття: значення, отримане на кроці 3, порушується модулем 2 із старшою половиною перетвореного блоку.

Крок6. Зсув по ланцюжку: Молодша частина перетвореного блоку зміщується на місце старшого, а на його місце розміщується результат попереднього кроку.

Тоді структура алгоритму може описати діаграму, представлену на рис. 3.8.

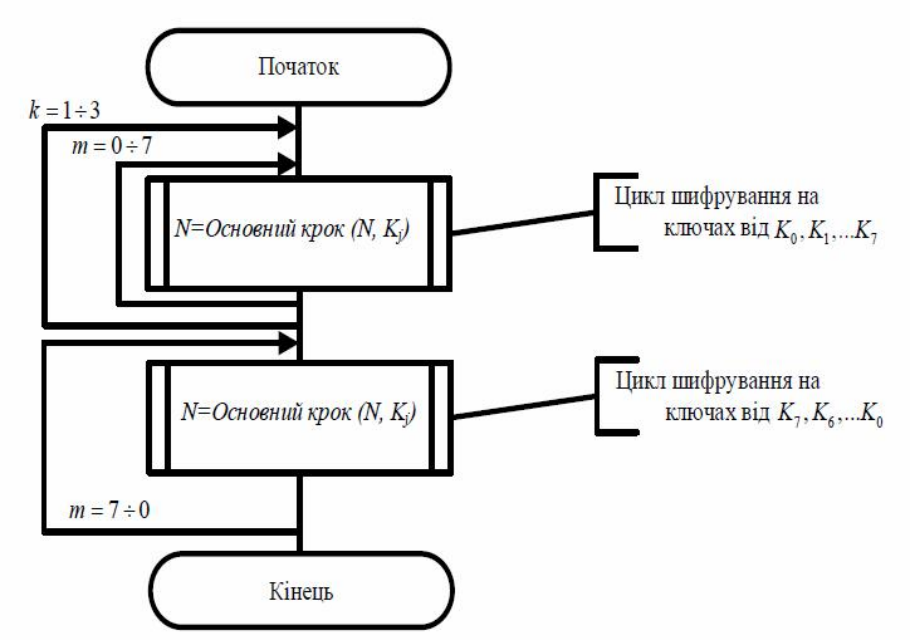

Рис. 3.8. Цикл шифрування

При передачі інформації від джерела поширення далі по ланцюжку можна спостерігати факт «загасання» або «спотворення» первинної інформації. Для дослідження цього феномена збір даних про об'єкти соціальної мережі повинен включати також збір відомостей про їх інформаційний простір. Схема алгоритму збору даних про відносини «репост» між об'єктами соціальної мережі приведена на рис. 3.9.

Для оцінки ступеня спотворення інформації в міру переміщення вихідного об'єкта за ланцюжком поширення необхідно піддати зібрані дані попередній обробці. Як механізм обробки текстового наповнення інформаційних об'єктів пропонується використовувати метод обчислення ключових слів, реалізований,

наприклад, в системі зі створення українськомовного корпусу для автоматичного перефразування і пошуку синонімів «ParaPhraser». Формально алгоритм попередньої обробки інформаційного простору об'єктів може бути записаний у такий спосіб. Нехай K(Q") - функція обчислення k ключових слів за інформаційним простором об'єкта u ∈ Z. Параметр k може мати довільні значення і задається разом з набором вхідних даних X перед початком дослідження. На першому кроці слід обрахувати ключові слова для всіх n.

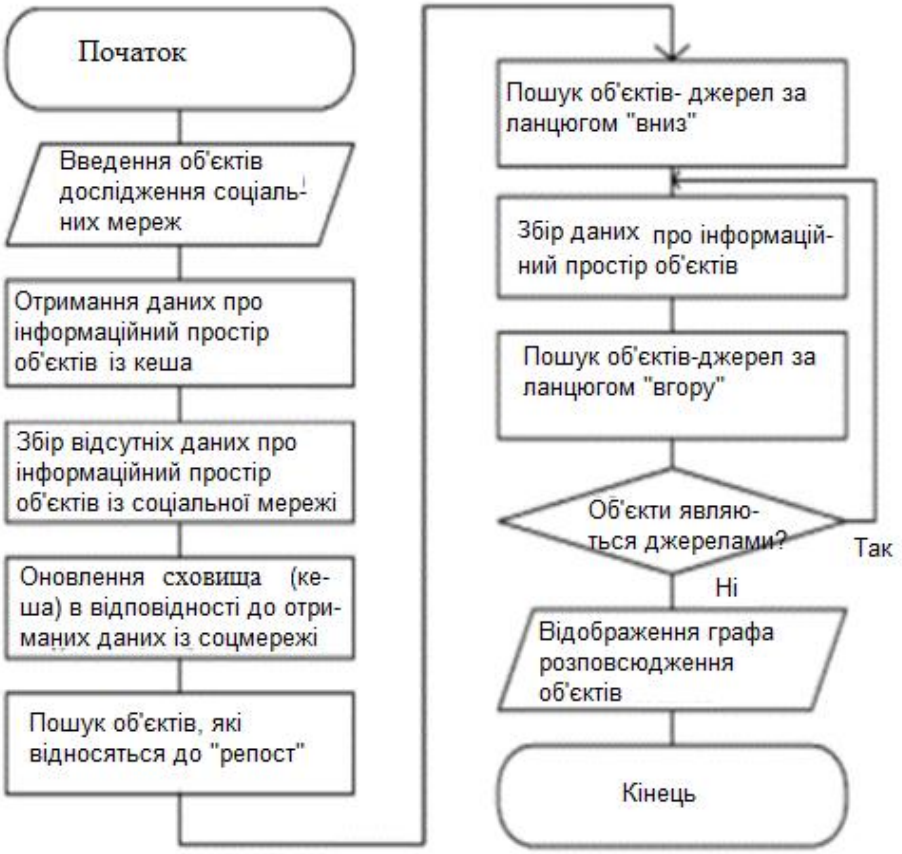

Рис. 3.9. Схема алгоритму збору даних

3.3 Визначення переваг методології забезпечення безпеки інформації в соціальних мережах

Методологічні основи забезпечення безпеки інформації в соціальних мережах являють собою сукупність концептуальних, теоретичних та технологічних основ. Концептуальні основи складаються з концептуальних

положень, визначень, принципів, поглядів та шляхів вирішення сформульованої наукової проблеми. Теоретичні основи складаються з математичних моделей, методів та методик, підпорядкованих вирішенню зазначеної наукової або науково–прикладної проблеми. Технологічні основи складаються з практичних методик, технологій, особливостей їх використання, а також практичних рекомендацій щодо реалізації теоретичних результатів та практичних методик.

Тому структура методологічних основ забезпечення безпеки інформації в соціальних мережах буде мати такий вигляд (рис. 3.10).

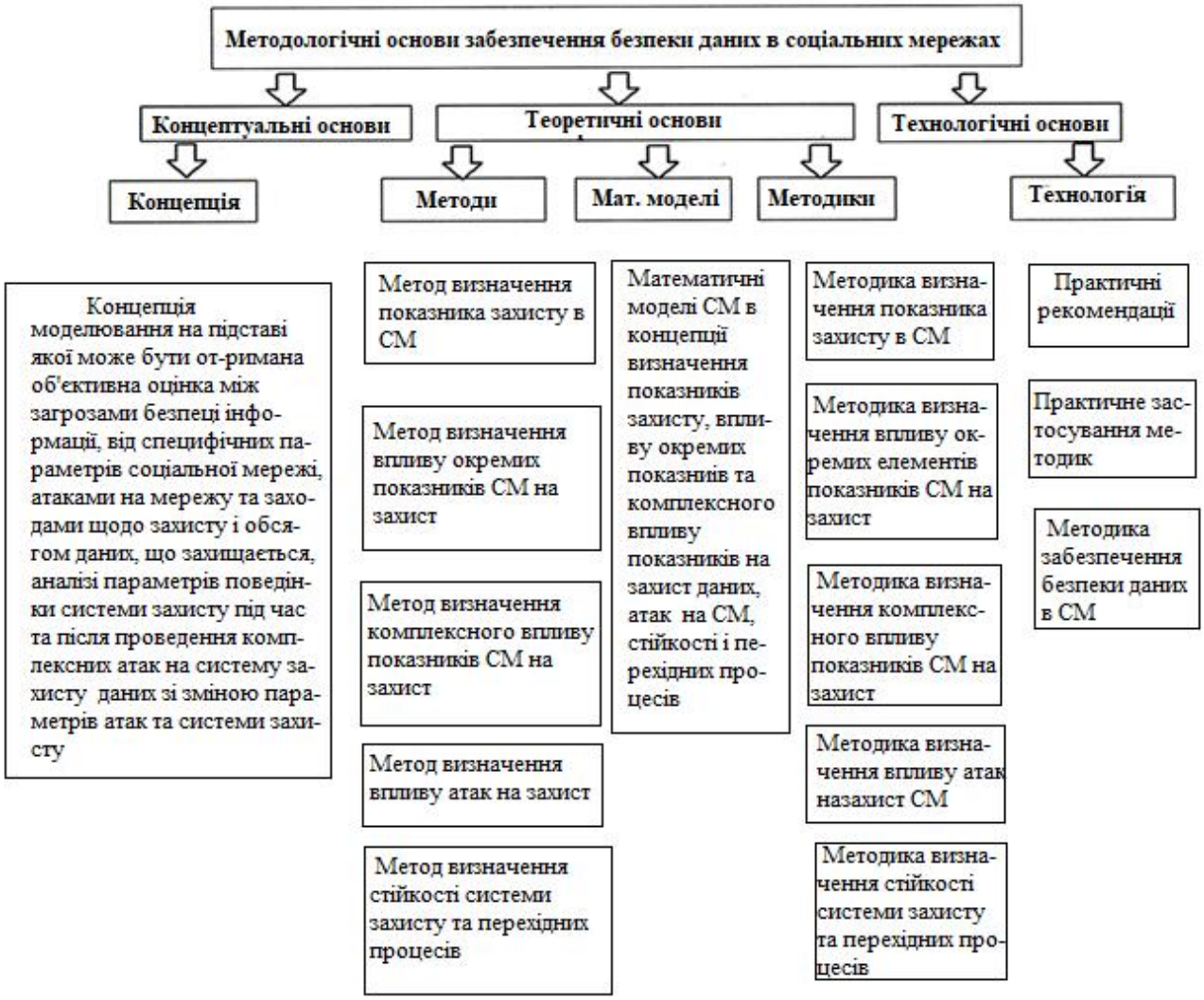

Рис. 3.10. Методологічні основи забезпечення безпеки інформації в соціальних мережах

Концептуальні положення визначають стратегічні шляхи удосконалення та розробки методологічних основ. Вони заклали основні напрямки розвитку методології забезпечення безпеки інформації в соціальних мережах. Основні переваги розроблених методологічних основ забезпечення безпеки інформації в соціальних мережах на шляху стратегічних напрямків розвитку безпеки інформації в СМ мають теоретичні основи які складаються з концепції, методів, математичних моделей та методик забезпечення безпеки інформації в соціальних мережах.

Тому для визначення переваг розберемо більш детальніше цей напрямок методологічних забезпечення безпеки інформації в соціальних мережах. Як показано, забезпечення безпеки інформації в соціальних мережах виконується «класичними методами» з використанням, маршрутизаторів, брандмауерів, антивірусного програмного забезпечення, підмереж захисту, методів доступу і т.п. не враховуючі при цьому впливу специфічних параметрів СМ (конфіденційність інформації, репутація користувачів мережі, взаємовплив користувачів, довіра між користувачами, спільні думки користувачів мережі, сильні та слабкі зв’язки, сила користувача, або центральність вузлів, швидкість поширення інформації в мережі, параметри розширення мережі, кількість співтовариств в мережі, канали поширення інформації, ідентифікація користувачів і т.п.), крім того оцінка стійкості системи захисту, перехідних процесів проводиться емпіричним шляхом, або акцент робиться на загальних аспектах і методах виявлення загроз та захисту інформації користувачів, більшість наукових досліджень мають дескриптивний характер які розглядаються окремо із різних математичних та технічних напрямків.

Одразу хочу відмітити, що практично, усі існуюче забезпечення безпеки інформації в соціальних мережах виконується без врахування специфічних параметрів СМ. Розроблені методологічні основи забезпечення безпеки інформації в соціальних мережах відразу набувають переваги за рахунок удосконалення методу захисту інформації, врахування дії кожного специфічного

параметра соціальної мережі та їх комплексної дії, а також впливу впливів зі зміною параметрів впливу та параметрів мереж, їх нелінійної взаємодії.

Визначення параметрів системи захисту інформації від дії параметрів самої мережі та параметрів впливу усовує недолік методу «класичного захисту».

Висновки до розділу

В даному розділі виконана реалізація методології контролю доступу та захисту статичної і динамічної інформації в соціальних мережах. Метод визначення шкідливих впливів на параметри захисту інформації в соціальних мережах, який, на відміну від існуючих, відрізняється застосуванням системи диференційних рівнянь, які враховують комплексні параметри нападу є наступною перевагою розроблених метологічних основ забезпечення безпеки інформації в соціальних мережах.

ВИСНОВКИ

В магістерській роботі досліджено моделі, методи та засоби контролю доступу до статичної та динамічної інформації та способи її захисту в умовах соціальних мереж.

Актуальність проблеми контролю доступу до інформації залишається актуальною через зростання обсягів даних та збільшення кількості кібер загроз у сучасному світі, особливо в соціальних мережах. Потреба у вдосконаленні - існуючі моделі та методи контролю доступу вимагають вдосконалення для врахування сучасних технологій, зростаючої динамічності інформації та нових викликів безпеки.

Основними результатами дослідження є розробка нових моделей та методів контролю доступу, спрямовані на забезпечення ефективного та безпечного управління інформацією. Результати дослідження включають інноваційні засоби та рекомендації для захисту інформаційних систем, зокрема соціальних мереж. Практичне значення полягає в тому, що отримані результати можуть бути застосовані у різних сферах, включаючи бізнес, охорону особистої інформації та інші галузі, де важлива безпека інформації. Порівняння з існуючими рішеннями - розроблена методологія демонструє переваги порівняно з існуючими підходами.

Дослідження підтверджує необхідність та можливість вдосконалення систем контролю доступу до статичної та динамічної інформації, забезпечуючи важливий внесок у розвиток кібербезпеки та захисту інформації у сучасному інформаційному суспільстві.

Таким чином розроблені методологічні основи забезпечення безпеки інформації в соціальних мережах у рамкам розробленої концепції перевершують існуючі методи та методики, які використовуються у сучасних соціальних мереж, наукових дослідженнях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ВОЛІН, С.; РУНЕСОН, П.; ХОСТ, М.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. Експерименти в розробці програмного забезпечення. [s.l.] Springer, 2012. стор. 250. 2012 рік.

2. BICCEP, E. WebDSL: тематичне дослідження предметно-орієнтованої мовної інженерії. У: Генеративні та трансформаційні методи в інженерії програмного забезпечення II. Брага, Португалія: Springer Berlin Heidelberg, 2008. стор. 291–373.

3. ФОЛТЕР, М. Архітектура як мова. Програмне забезпечення IEEE, v. 27, n. 2, стор. 56 – 64, 2010.a.

4. VOELTER, M. Розробка вбудованого програмного забезпечення з робочими місцями проекційної мови. Матеріали 13-ї міжнародної конференції з інженерних мов і систем, керованих моделями, частина II (MoDELS 2010). Anais... : MODELS'10.Springer-Verlag, 2010.b.

5. ФЕЛЬТЕР, М.; BICCEP, E. Розширення мови та композиція за допомогою мовних робочих місць. Матеріали супутньої міжнародної конференції ACM на тему «Об'єкт орієнтовані мови систем програмування та компаньйон додатків– SPLASH10

Anais... Нью-Йорк, Нью-Йорк, США: ACM Press, 17 вихід. 2010 рік.

6. ВАНГ, Б.; WEN, C.; ЖУ, B.; SHENG, J. Awright-ADL for Aspectual Component Composition Platform. 2008 Міжнародний симпозіум з комп'ютерних наук і обчислювальних технологій. Анаіс... Ieee, 2008.

7. WANG, Q. Y.; ІНГ, С.; JIA, X. Y.; Л. В., Г. БІН; SHUAI, Y. SOADL-EH: Мова опису сервіс-орієнтованої архітектури, що підтримує обробку винятків. Advanced Materials Research, т. 433-440, с. 3500–3509, січ. 2012 рік.

8. ВАСУДЕВАН, Н.; TRATT, L. Порівняльне дослідження інструментів DSL. Електронні нотатки в Теоретична інформатика, v. 264, n. 5, стор. 103–121, лип. 2011 рік.

9. УШАХІДІ. Ушахіді. 2014. Опубліковано: <http://www.ushahidi.com

10. ВАЙДА, А.; EKER, J. Повернення до мовного лісу. Матеріали семінару FSE/SDP щодо майбутнього дослідження програмної інженерії - FoSER ’10. Anais... Нью-Йорк, Нью-Йорк, США: ACM Press, 2010.

11. РУШКОФФ Д. Програмуй або будь запрограмованим: десять команд для цифрової ери. BetterListen, 2013. 2013.

12. САРІМБЕКОВ А.; ПОДЗІМЕК, А.; БУЛЕЙ, Л.; ЧЖЕН, Ю.; РІЧІ, Н.; БІНДЕР, В. Характеристики динамічних мов JVM. Матеріали 7-го семінару ACM з віртуальних машин і проміжних мов - VMIL ’13. Анаіс... Нью-Йорк, Нью-Йорк, США: ACM Press, 2013.

13. СОЙЄР, П.; БЕНКОМО, Н.; Х'ЮЗ, Д.; GRACE, Р.; ГОЛДСБІ, Г. Дж.; CHENG, B. H. C. Візуалізація аналізу динамічно адаптивних систем за допомогою i* та DSL. Другий міжнародний семінар з інженерної візуалізації вимог REV.

14. Anais... IEEE Computer Society, 2007. ПРЕСМАН, Р. С. Розробка програмного забезпечення: практичний підхід. 7. вид. Нью-Йорк, Нью-Йорк, США: McGraw-Hill, Inc., 2009. стор. 928. 2009 рік.

15. ПРОГРАМОВАНИЙ ВЕБ. API як конкурентна перевага. 2013. Доступно: http://blog.programmableweb.com/2013/01/09/apis-as-a-competitive-advantage/>.

16. ПРОГРАМОВАНИЙ ВЕБ. ProgrammableWeb – Мешапи, API та Інтернет як платформа. 2014. Доступно: <http://www.programmableweb.com/>.

17. РАЦІОНАЛЬНІ ПАРТНЕРИ. Семантика UML v1.1. 1997. Disponível em: <http://www.omg.org/cgi-bin/doc?ad/97-08-04.pdf>.

18. RESNICK, M. та ін. Scratch: програмування для всіх. повідомлення ACM, v. 52, n. 11, стор. 60, 1 лист. 2009 рік.

19. РІЧАРДСОН, Л.; АМУНДСЕН, М.; RUBY, S. RESTful Web API. [s.l.] O’Reilly Media, 2013. стор. 379. 2013 рік.

20. NG, J. W. Від репрезентативної передачі стану (REST) до репрезентативної передачі стану дії (REAST): увімкнення розумніших веб-взаємодій для веб-задач. Матеріали конференції Центру перспективних

досліджень спільного дослідження 2013 р. Anais... : CASCON ’13.Riverton, NJ, USA: IBM Corp.,2013.

21. PANDEY, R. K. Архітектурні мови опису (ADL) проти UML: огляд. ACM SIGSOFT Software Engineering Notes, v. 35, n. 3, стор. 1, 11 травня 2010 р.

22. NASCIMENTO, L. M. та ін. Систематичне картографічне дослідження предметно-специфічного мови. Сьома міжнародна конференція з прогресу програмної інженерії (ICSEA 2012). Anais... Лісабон, Португалія: 2012.

23. МІХЕЛС, С.; PLASMEIJER, R. iTask як нова парадигма для створення GUI програми. Матеріали 22-ї міжнародної конференції з імплементації та застосування функціональних мов (IFL’10). Anais... Springer, 2010.

24. МОХАММАД, М.; ALAGAR, V. TADL - Мова опису архітектури для надійні компонентні системи. Proceedings of ECSA ’08 Proceedings of the 2-га Європейська конференція з архітектури програмного забезпечення. Анаіс... 2008 рік.

25. МОРЕНО-ГЕР, П.; ФУЕНТЕС-ФЕРНАНДЕЗ, Р.; СЬЄРРА-РОДРІГЕС, Ж.-Л.; FERNÁNDEZ-MANJÓN, В. Перевірка моделі для пригодницьких відеоігор. Інформація and Software Technology, v. 51, n. 3, стор. 564–580, 2009.

26. МЕРКЛ, Б. Інструменти текстового моделювання: огляд і порівняння мови верстаки. Матеріали супутньої міжнародної конференції ACM на тему «Об’єкт орієнтовані мови систем програмування та компаньйон додатків - SPLASH ’10. Anais... Нью-Йорк, Нью-Йорк, США: ACM Press, 17 вихід. 2010 рік.

27. MEIRA, S. R. L. та ін. Нова мережа соціальних машин. 2010. Disponível em:<http://arxiv.org/ftp/arxiv/papers/1010/1010.3045.pdf>. Доступ до: липень 2012 р.

28. MEIRA, S. R. L. та ін. Нова мережа соціальних машин. 2011 35-й щорічний IEEE Конференція з програмного забезпечення та прикладних програм. Анаіс... IEEE, лип. 2011 рік.

29. МАНДЖУНАТА, А.; РАНАБАХУ, А.; ШЕТ, А.; ТІРУНАРАЯН, К. Сила Хмари у вашій кишені: ефективний підхід для Cloud Mobile Hybrid Розробка додатків. 2010 Друга міжнародна конференція IEEE з хмар Обчислювальна техніка та наука. Анаіс... Ieee, 2010.

30. МАРИНО, Дж.; РОУЛІ, М. Розуміння SCA (архітектура сервісних компонентів). [s.l.] Addison-Wesley Professional, 2009. стор. 360. 2009 рік.

31. МАРТІН ГУДГІН та ін. Специфікація W3C SOAP V. 1.2. 2007. Disponível em:<http://www.w3.org/TR/soap12>. Доступ до: Abr. 2012 рік.

32. МАРТІНЬО, Р.; ВАРАДЖО, Дж.; DOMINGOS, D. Використання семантичної мережі для визначення а мова для моделювання керованої гнучкості програмних процесів. Програмне забезпечення IET, v. 4, n. 6, стор. 396, 2010.

33. LINDEMAN, R. T.; КАЦ, Л. К. Л.; BICCEP, E. Декларативне визначення предметно-спеціальних мовних налагоджувачів. Матеріали 10 міжнародної конференції ACM на Генеративне програмування та розробка компонентів - GPCE ’11. Анаіс... Нью-Йорк, Нью-Йорк, США: ACM Press, 2011.

34. ЛОХМАНН, Х.; HESSELLUND, A. Інтегрований погляд на моделювання з кількома предметно-орієнтовані мови. Матеріали міжнародної конференції IASTED Software Engineering SE 2009. Anais... ACTA Press, 2009.

35. ЛОПЕС-САНЦ, М.; КУЕСТА, С. Е.; MARCOS, E. Формалізація високорівневих сервіс-орієнтованих архітектурних моделей за допомогою динамічного ADL. OTM’10 Матеріали 2010 року міжнародна конференція про перехід до значущих систем Інтернету. Анаіс... : 9-е Зізнався. Міжн. Семіни та плакати на тему «На шляху до значущих Інтернет-систем»,

36. OTM 2010: AVYTAT 2010, ADI 2010, DATAVIEW 2010, EI2N 2010, ISDE 2010, MONET 2010, OnToContent 2010, ORM 2010, P2P-CDVE 2010, SeDeS 2010, SWWS 2010 та OTMA 2010.2010.

37. КРАН, Г.; РУМПЕ, Б.; VÖLKEL, S. MontiCore: основа для композиції розробка предметно-спеціальних мов. Міжнародний журнал про програмні засоби для Передача технологій, т. 12, п. 5, стор. 353–372, 2010.

38. KRÄMER, B. J. Компонент зустрічається з обслуговуванням: як виглядає дворняжка? Інновації у системах і розробці програмного забезпечення, т. 4, п. 4, стор. 385–394, 13 лист. 2008. KASSEM, R.; НАВІСЕНИЙ, М.; BÉCHENNEC, J.-L.; САВАТОН, Г.; ТРІНКЕТ, Ю. Harmless, мова опису апаратної архітектури, присвячена вбудованим у режимі реального часу системне моделювання. Журнал системної архітектури, v. 58, n. 8, стор. 318–337, упоряд. 2012 рік.

39. КАЦ, Л. К. Л.; ВІССЕР, Е. The spoofax language workbench. ACM SIGPLANПовідомлення, т. 45, п. 10, стор. 444, 17 вихід. 2010 рік.

40. КАУФМАНН, М.; ХЕСІНГ, М.; ПРЕУСС, Т.; СПАЛЛЕК, Р. Віртуальна Java Машина в моделюванні високопродуктивного набору інструкцій з можливістю перенацілювання. Праці 9-ї Міжнародної конференції з принципів і практики програмування на Java - PPPJ ’11. Anais... Нью-Йорк, Нью-Йорк, США: ACM Press, 2011.

41. КЕННЕДІ, К. та ін. Телескопічні мови: система автоматичної генерації Мови домену. Праці IEEE, v. 93, n. 2, стор. 387–408, лют. 2005 рік.

42. КІЧЕНХЕМ, Б. Рекомендації щодо виконання систематичних оглядів літератури в програмна інженерія, версія 2.3. Технічний звіт EBSE університету Кіл EBSE200701. Велика Британія: Кільський університет, технічний звіт EBSE. EBSE-2007- 01, 2007.

43. KLEIN, P. Модель мови опису архітектури. В: ENGELS, G.; LEWERENTZ, C.; ШЕФЕР, В.; ШУРР, А.; WESTFECHTEL, В. (Ред.). Графік Перетворення та модельно-керована інженерія. Конспект лекцій з інформатики. Берлін, Німеччина: Springer Berlin / Heidelberg, 2010. v. 5765p. 249–273.

44. КОСАР Т. та ін. Порівняння мов загального призначення та предметноспеціальних мов: емпіричне дослідження вивчення. Комп'ютерні науки та інформаційні системи, т. 7, н. 2, стор. 247–264, 2010.