УДК.004.415

к.т.н., доц. **Муляр І.В.** (ХмНУ)
к.т.н., с.н.с. **Лоза В.М.** (ВІКНУ)
Войнарович С.Б. (ХмНУ)

АНАЛІЗ ПІДХОДІВ ДО СТРУКТУРНОЇ ЗБІРКИ WEB-ДОДАТКІВ

У статті розглядається автоматизована компонентна збірка Web-додатків. Існує велика кількість готових рішень по компонентній збірці, але такі рішення мають ряд недоліків, які хотілося б мінімізувати під час побудови власного проекту. Основним завданням дослідження є створення моделі для компонентної збірки, яка буде базуватись на семантичній моделі.

У процесі роботи пропонується розглянути компоненту збірку, як інформаційну систему. Також велика увага приділяється структурному синтезу. Приводяться приклади представлення результатів структурного синтезу даних. Виконано опис основних методів структурного синтезу та їх графічні схеми. Слід виділити три основні рівні компонентної збірки: процесний, компонентний та фізичний. Після аналізу було запропонованно ряд логічних припущень, які згодом будуть використовуватись у створені семантичної моделі. Було взято семантичну модель, що використовується як основа для побудови бази знань онтологічного типу.

Основним рішенням у ств оренні майбутньої системи автоматизованої збірки є розбиття трьох окремих систем на підсистеми, які успішно взаємодіють одна з одною. Така система має доступ до бази даних, яка постійно поповнюється вхідними даними. Вхідні дані проходять етап аналізу та форматування і у будь-який час можуть викликатись, як готові рішення для компонентної збірки. Звертання до бази даних відбувається у вигляді запитів.

Ключові слова: Web-додаток, семантична модель, компонентна збірка, автоматизована система.

Вступ. У наш час важко уявити будь-яку організацію, магазин чи навчальний заклад, який не має власного веб-сайту. Велика кількість замовлень припадає на сферу web-розробки, а саме написання web-додатків. Станом на 2016 рік одну із лідируючих позицій серед мов програмування, що використовуються для написання функціональної частини web-додатку (back-end part) займає Java. Збірка таких проектів складається із багатьох функціональних компонентів, тому такий процес називається компонентною збіркою web-додатків. У мові програмування Java існує велика кількість уже готових рішень для автоматизації збірки

проектів. Найпопулярнішими рішеннями є Grandle, Ant та Maven. Основним завданням цих інструментів є виконання користувацьких інструкцій та перетворення існуючих програмних модулів у архіви із заданою структурою, автоматизація опису та встановлення залежностей. Існує і великий ряд недоліків та проблем під час використання готових рішень для збірки. Зустрічаються випадки, коли при написанні не складного web-додатку використовувалось до 50-60 відсотків незадіяних модулів та бібліотек, тому користувач або отримує громісткий проект, або втрачає дорогоцінний час на індивідуальну підбірку компонентів для свого проекту.

Постановка завдання. З метою покращення збірки web-додатків розробники намагаються знайти рішення, яке дозволить максимально ефективно використовувати існуючу компонентну базу, що збільшить швидкість збірки та полегшить редагування проекту на пізніх етапах його створення.

Однак кожний продукт має власний метод опису залежностей для подальшої компонентної збірки проекту. Це суттєво ускладнює пошук та розробку оптимального рішення. Наприклад, для Maven залежності збірки описуються у спеціальних файлах-конфігураторах, які виступають набором атрибутів для подальшого використання описаних модулів, бібліотек, артефактів та актуальних версій. Потім середовище використовує файли-конфігуратори, які, слідуючи інструкціям, виконують пошук і завантаження у проект потрібних компонентів. Відсутність у проекті важливих для його функціонування компонентів може призвести до некоректної роботи, або взагалі до непрацездатності. Існує велика кількість популярних онлайн-репозиторіїв, які можуть запропонувати готові рішення для збірки, але вони потребують досить чітких знань та навичок, що призводить до втрати часу на їх вивчення.

Отже, розроблювальне рішення для покращення збірки проектів можна вважати, як додатковий етап серед вже існуючих. Рішення підтримки збірки проектів (РПЗП) потребує чіткого визначення, опису та категоризації компонентів і модулів за допомогою семантичних мереж. Використання таких мереж дозволить провести чіткий зв'язок між поставленими задачами та оптимальними рішеннями. Головна ідея - це сприйняття кожного логічного процесу, як окрему задачу, а програму, як існуюче рішення. Дотримуючись такої схеми, можна прискорити пошук рішень для конкретних задач.

Виклад основного матеріалу дослідження. Існує ряд уже готових рішень по автомазованій збірці Web додатків, але вони мають свої недоліки та переваги. Найпопулярнішим рішенням вважається Maven[1]. Apache Maven – це засіб автоматизації роботи з програмними проектами, який спочатку використовувався для Java проектів, який використовується для управління (management) та складання (build) програм. Створений 2002 року Джейсоном ван Зилом. За принципами роботи кардинально відрізняється від Apache Ant, та має простіший вигляд щодо build-налаштувань, яке надається в форматі XML. XML-файл описує проект, його зв'язки з зовнішніми модулями і компонентами, порядок будування (build), папки та необхідні плагіни. Основним плюсом є те, що база із додатковими модулями та додатковими бібліотеками розміщується на серверах. Наступним у списку є Ant. Ant – платформонезалежний аналог UNIX-утиліти make, але з використанням мови Java, він вимагає платформи Java, і краще пристосований для Java-проектів. Grandle є натупною системою, яка наслідує принципи закладені у Maven та Ant. Grandle використовує предметно-орієнтовану мову (DSL) на основі мови Groovy замість традиційної XML-подібної форми представлення конфігурації проекту, що являється його перевагою. Для визначення порядку виконання завдань Gradle використовує орієнтований ациклічний граф ("DAG"). Звісно такі готові рішення користуються великою популярністю серед розробників на мові Java, але існуючі недоліки спонукають до розробки більш збалансованої системи автоматичної збірки Web-додатків.

Web-додаток можна сміливо віднести до інформаційної системи (ІС). Тому представлену ІС можна розглядати, як сукупність компонентів та зв'язків між ними. Важливим етапом на шляху до розробки системи компонентної збірки web-додатків є структурний синтез [2]. Під

час такого синтезу потрібно провести чіткий опис усіх компонентів, що входять до складу ІС та встановити зв'язки між ними. Представлення результатів структурного синтезу може відбуватися такими способами:

- граф зв'язків;
- блок-схема;
- структурна схема;
- креслення;
- ескіз.

Окрім предсталення структурного синтезу існує і ряд математично-логічних методів, які орієнтовані на вирішення задач компонентної збірки [2]. Слід розглянути особливості таких методів:

1. Технічна система має деяку чітку структуру.
2. Технічна система належить до класу об'єктів, що мають однакові функціональні призначення.
3. Сукупність аналогів та прототипів має здатність до нових результативних комбінацій.

Слід розглянути основні методи, які в подальшому можуть використовуватись для побудови семантичних мереж з використанням структурного синтезу.

Структурний синтез по альтернативних деревах [3]. Такий метод містить декомпозиції функцій системи.

A-деревом (рис. 1) називається вектор (G, S, γ), де:

а) $G = G(W, D)$ - дерево із підмножиною вершин $W = \{w_i\}$ і підмножиною дуг $D = \{d_j\}$;

б) $S = \{S_k\}$ - підмножина номерів зв'язків;

в) $\gamma S \to 2^D$ приводить у відповідність кожному імені зв'язки s, яка належить S, підмножину дуг $\gamma(s)$ із множини S, при цьому всі дуги однієї зв'язки мають загальну початкову вершину.

Метод синтезу с використанням N-дольних графів дозволяє групувати функції иехнічного за об'єкта за їх реалізацією.

N-дольний граф - це граф виду $G = (X_b X_2 ... X_n, R)$, де $X = UX$ - множина вершин, R - множина ребер. Підмножину X прийнято називати долями. Ребра можуть з'єднувати лише вершини . Вершини, які належать одній долі завжди є незалежними підмножинами.

Метод морфологічного синтезу . Цей спосіб дозволяє проводити систематизацію та пошук усіх можливих способів побудови об'єкта, який має функціональне призначення. Для опису можливих комбінацій використовується матриця інцидентності. Загальна схема має такий вигляд:

1. Визначити функції, які допустимий об'єкт зможе виконати.
2. Перерахувати спектр часткових рішень (альтернативні засоби виконання кожної функції).
3. Вибрати по одному допустимому рішенні для кожної системи.

Обмеження:

1. Зв'язки дерева описують способи розбиття функцій на підфункції.
2. Висячі вершини (листи) представляють технічну реалізацію.
3. Корінь дерева відповідає основній технічній функції класа об'єкта.

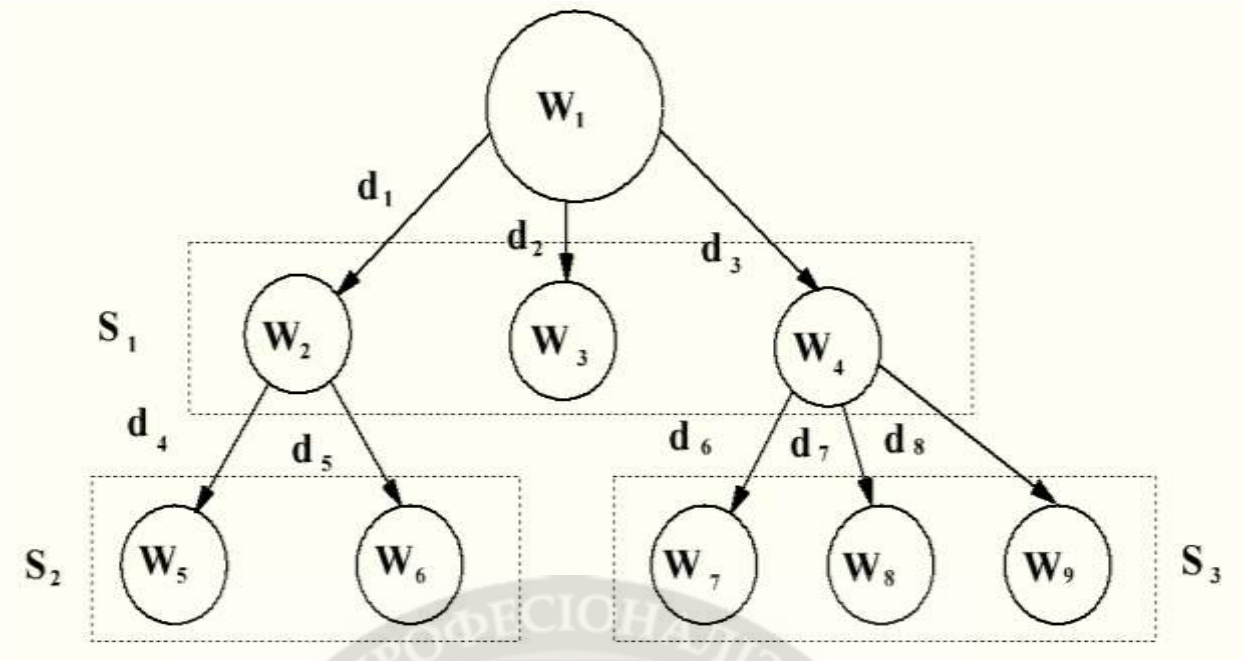

Рис. 1. Граф, який використовується для вирішення задачі структурного синтезу

Існують інші методи компоненної зібрки на основі графових моделей, але вони є модифікаціями розглянутих нами методів.

Виділено три основні стадії компонентної збірки Web-додатків [2]:

- процесна (рівень загальної структури);
- компонентна (рівень елементів структури);
- фізична (рівень файлів Web-додатку);

Будь-яка ІС використовує стандартизовані профілі [4], які характеризиються:

- спеціальними особливлстями процесів предметних областей,
- профілями уже існуючих ІС, які використовуються в якості компонентів.

У нашому випадку не існує чіткої предметної області, тому можна використовувати ІС, які будуть вирішувати однакові завдання, але будуть мати різну архітектуру. Такі системи будуть відрізнятися і за профілями. Альтернативні процеси і компоненти ІС базуються на інформаційних і телекомунікаційних структурах [5]. Використання таких структур у своїй основі - це правильне рішення, оскільки вони постійно вдосконалюються і використовують найновіші технічні рішення. Такі рішення без особливих проблем інтегруються у ІС і надають їм актуальності, що призводить до постійної конкуренції та підштовхують до пошуку максимально продуктивних рішень.

На основі розглянутих методів можна отримати ряд логічних припущень, які згодом будуть втіленні у семантичну модель:

- сумісність компонентів визначається на основі міжпроцесних потоків та множини компонентів із зовнішніх інтерфейсів, які мають власні обмеження?
- множина процесів та компонентів має дуже велику потужність.
- деякі компоненти та їх повний опис доступні у мережі Інтернет.
- для web-додатку не може бути сформована кінцева структура, оскільки кількість його компонентів постійно змінюється.

Потреба використання у проекті семантичних моделей базується на тому, що вони дозволяють формалізувати складні предметні області. Взявши за основу семантичну модель, можна сформувати середовище стандартизації. У нашому випадку семантична модель буде використовуватись для побудови інформаційного середовища, що дозволить отримати динамічну систему. Така система буде працювати із введенням та опрацювання інформації про існуючі компоненти, які будуть використовуватись для автоматизованої компонентної збірки Web-додатків.

Семантичні моделі є основою для побудови бази знань онтологічного типу, який визначається за формулою:

$$S^d = \{O, M, R\},$$

де O - набір взаємопов'язаних онтологій, які об'єднують класифікатори, сервіси, апаратні системи та їх специфікації. M - семантичні метадані, R - множина правил (логічних). Представлена загальна схема майбутньої автоматизованої системи компонентної збірки Web-додатків. В даній схемі комбінація систем представлена у вигляді підсистеми первинного аналізу і форматування даних, підсистеми, яка формує кінцевий склад Web-додатків та підсистема управління збереження даних про процеси та компоненти (рис. 2).

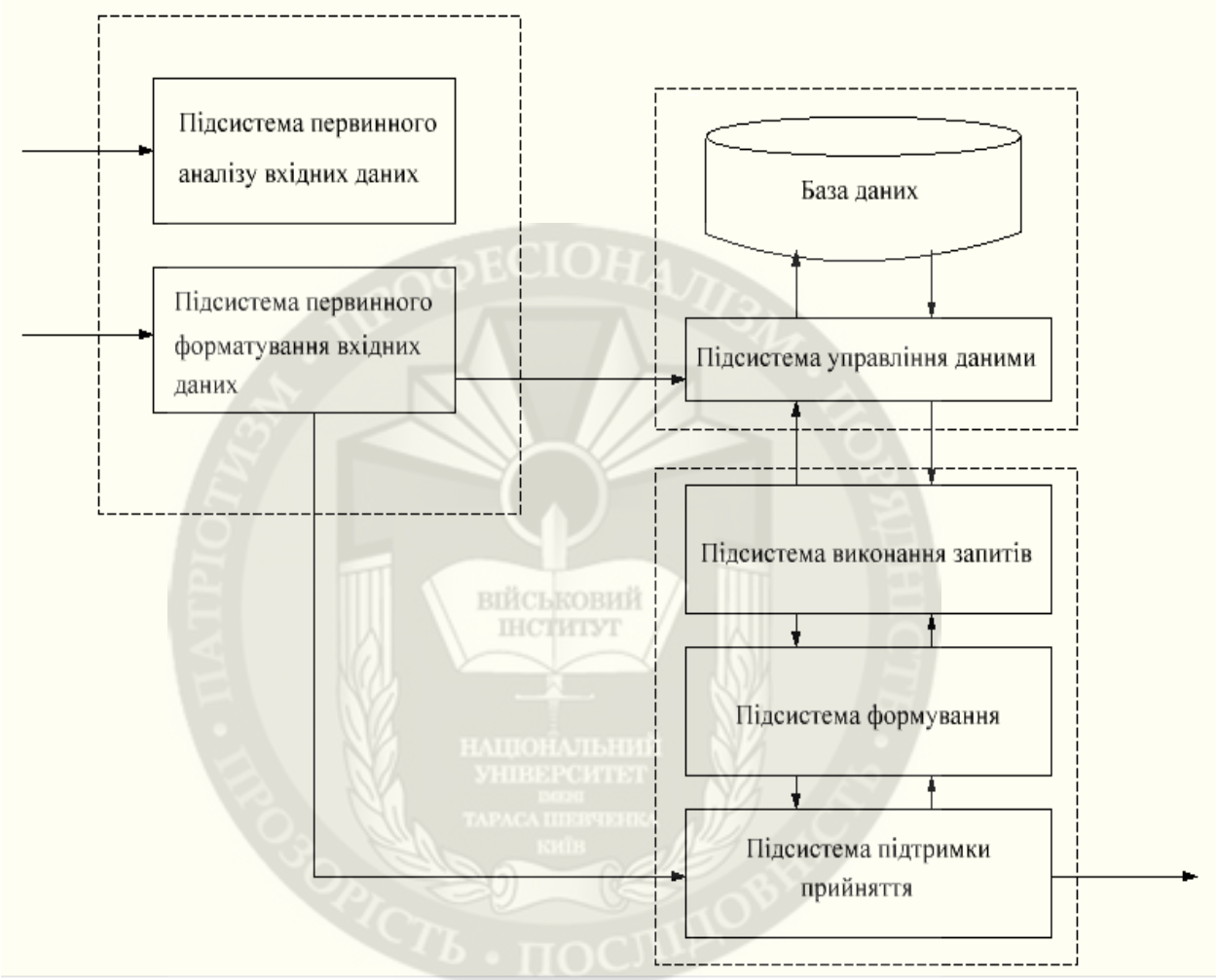

Рис. 2. Загальна схема автоматизованої системи компонентної збірки Web-додатків

Семантична модель, побудована на такій схемі, повинна враховувати і оцінювати індивідуальні властивості компонентів і процесів, які входять до неї. Також потрібно слідувати ієрархії процесів та зберігати списки готових рішень, які успішно застосовуються. Доцільним є використання компонентів, які реалізують процеси, що можуть взаємодіяти один з одним, використовуючи свої інтерфейси. Така взаємодія призводить до прискорення компонентної збірки.

Беручи за основу схема (рис. 2), було зроблено ряд припущень, відносно майбутньої системи. Підбірка процесів буде відбуватися ітераційно і на кожному кроці буде формуватись множина процесів, які будуть претендувати на потрапляння у структури Web-додатку. Компоненти можна умовно розділити на зовнішіні та внутрішні.

Зовнішні компоненти надають системі гнучкості, оскільки вона може використовувати компоненти в режимі віддаленого доступу, використовуючи онлайн ресурси. При наявності успішного підключення до мережі Інтернет, можна завантажувати компоненти, які

знадобилися системі для автоматичної збірки. Внутрішні компоненти потребують більшої уваги тому, що їх не можна завантажити онлайн. Такі компонети додаються вручну під час формування збірки. Вони представлені у вигляді різноманітних класів та бібліотек.

Висновки. У статті розглянуто автоматизовану компонентну збірку Web-додатків. Кінцевими продуктами таких збірок є додатки, які відносяться до інформаційних систем (ІС), тому їм приділенно достатньо уваги. До компонентів ІС застосовується структурний синтез. Було розглянуто способи представлення такого синтезу та ряд математично-логічних моделей, які орієнтовані на вирішення задач компонентної збірки. На основі розглянутих методів було зроблено ряд логічних припущень, які згодом будуть використовуватись у побудові семантичної моделі. Також було побудовано загальну схему автоматизованої системи компонентної збірки Web-додатків. Така схема реалізує в собі розбиття систем на підсистеми, що суттєво спрощує розробку. Беручи за основу майбутньої системи представлену схему, ми сформували ряд функціональних потреб, які повинен реалізувати кінцевий продукт.

ЛІТЕРАТУРА:

1. Apache Maven Project. Project information [Електроний ресурс], http://maven.apache.org/project-info.html

2. Аристов А.В. Разработка и исследование алгоритмов компонентной сборки Web-приложений на основе семантических сетей: дис. д-ра техн. наук: 05.13.11 / Аристов Алексей Владиславович. - К., 2016. - 125 с.

3. Глазунов В.Н. Поиск принципов действия технических систем М.: 1990

4.Вендров, А. М. Проектирование программного обеспечения экономических информационных систем [Текст]: Учебник / А. М. Вендров. — М: “Финансы и статистика” – 2014. – 544 с

5. Грекул, В.И. Проектирование информационных систем / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина – 2-е изд., испр. — Интернет-университет информационных технологий - ИНТУИТ.ру, 2010. – 200 с.

REFERENCES:

1. Apache Maven Project. Project information [Elektronij resurs], http://maven.apache.org/project-info.html

2. Aristov A.V. Razrabotka i issledovanie algoritmov komponentnoj sborki Web-prilozhenij na osnove semanticheskih setej: dis. d-ra tekhn. nauk: 05.13.11, / Aristov Aleksej Vladislavovich. - K., 2016. - 125 s.

3. Glazunov V.N. Poisk principov dejstviya tekhnicheskih sistem M.: 1990

4. Vendrov, A. M. Proektirovanie programmnogo obespecheniya ehkonomicheskih informacionnyh sistem [Tekst]: Uchebnik / A. M. Vendrov. – M: “Finansy i statistika”, 2014. – 544 s

5. Grekul, V.I. Proektirovanie informacionnyh sistem / V.I. Grekul, G.N. Denishchenko, N.L. Korovkina – 2-e izd., ispr. – Internet-universitet informacionnyh tekhnologij - INTUIT.ru, 2010. – 200 s.

Рецензент: д.т.н., проф. Лєнков С.В., начальник науково-дослідного центру Військового інституту Київського національного університету імені Тараса Шевченка

к.т.н., доц. Муляр И.В., к.т.н., с.н.с. Лоза В.Н., Войнарович С.Б.

АНАЛИЗ ПОДХОДОВ К СТРУКТУРНОЙ СБОРКЕ WEB-ПРИЛОЖЕНИЙ

В статье рассматривается автоматизированная компонентная сборка Web-приложений. Существует большое количество готовых решений по компонентной сборке, но такие решения имеют ряд недостатков, которые хотелось бы минимизировать во время построения собственного проекта. Основной задачей исследования является создание модели для компонентной сборки, которая будет базироваться на семантической модели.

В процессе работы предлагается рассмотреть компонентную сборку, как информационную систему. Также большое внимание уделяется структурному синтезу. Приводятся примеры представления результатов структурного синтеза данных. Выполнено описание основных методов структурного синтеза и их графические схемы. Следует выделить три основных уровня компонентной сборки: процессный, компонентный и физический. После анализа было

предложено ряд логических допущений, которые впоследствии будут использоваться в создании семантической модели. Используемая семантическая модель является основой для построения базы знаний онтологического типа. Основным решением в создании будущей системы автоматизированной сборки является разбиение трех отдельных систем на подсистемы, которые успешно взаимодействуют друг с другом. Такая система имеет доступ к базе данных, которая постоянно пополняется входными данными. Входные данные проходят этап анализа и форматирования и в любое время могут привлекаться, как готовые решения для компонентной сборки. Обращение к базе данных происходит в виде запросов.

Ключевые слова: Web-приложение, семантическая модель, компонентная сборка, автоматизированная система.

Ph.D. Mulyar I.V., Ph.D. Loza V.N., Voinarovych S.B.

ANALYSIS APPROACHES TO THE STRUCTURAL BUILD OF WEB-APPLICATIONS

The article discusses the automatic component build for Web applications. There are a large number of ready-made solutions for component build, but such solutions have several drawbacks. that I would like to minimize during the construction of my own project. The main point of the research is the creation of a model for a component build. It will be based on the semantic model.

During the research, it is proposed to consider the collection as an information system. Attention is also paid to the structural synthesis. There are given examples of presentation results of structural synthesis of data. A description of the basic methods of structural synthesis and their graphic schemes are done. It is worth to note three main levels of component build: processual, componental and physical. Analysis offers a number of logical assumptions, which will be used in creating the semantic model. Semantic model was taken as the basis for creating a knowledge base of ontological type.

The main solution in creating automatic build system is dividing of three separate systems into subsystems that interact efficiently with each other. Such a system has access to database, which is updated with the input data frequently. The input data passes the analysis phase and formatting as well At any time it can be used as ready solutions for component build. Access to database occurs in the form of queries.

Keywords: web application, semantic model, component build, automatic system.