УДК 004.8
DOI: 10.15587/2312-8372.2019.183201

РОЗРОБКА МЕТОДУ ГЕНЕРАЦІЇ ВЕБ-КОНТЕНТУ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

Бібічков І. Є., Сокол В. В., Шевченко О. Ю.

РАЗРАБОТКА МЕТОДА ГЕНЕРАЦИИ ВЕБ-КОНТЕНТА НА ОСНОВЕ ОНТОЛОГИЧЕСКОГО ПОДХОДА

Бибичков И. Е., Сокол В. В., Шевченко А. Ю.

ONTOLOGICAL APPROACH TO DEVELOPMENT OF WEB-CONTENT GENERATION METHOD

Bibichkov I., Sokol V., Shevchenko O.

Об'єктом дослідження є процес автоматизованого створення веб-контенту, що базується на інформації, представленій в онтологічному вигляді. Одним із найбільш проблемних місць у веб-розробці є процес створення інтерфейсу користувача. Це пов'язано з тим, що даний процес є комплексним та вимагає витрати найбільшої кількості часу та коштів у порівнянні з іншими етапами розробки.

В ході дослідження було застосовано модель розробки програмного забезпечення, що базується на розробці онтології, а потім програмного застосування для її обробки. Даний підхід називається «Ontology-driven development» (або процес розробки програмного забезпечення (ПЗ), що керований онтологією).

Отримано інтелектуальну модель для представлення елементів веб-ресурсів у вигляді онтології, а також програмне ядро системи для генерації веб-сторінок на основі інформації, що зберігається в онтології. Це пов'язано із необхідністю отримання набору кінцевих елементів інтерфейсу користувача (HTML, CSS, JS елементів), із яких формуються веб-сторінки.

Налагодження адресації між сторінками веб-ресурсу має ряд особливостей, зокрема, для зв'язування адреси кінцевої веб-сторінки із контролером, що відповідає за генерацію її змісту, було запропоновано відповідний підхід. Цей підхід функціонує аналогічно так званому «роутеру», який використовується в класичних системах (наприклад, JSP для Java). Відмінною особливістю даного підходу є те, що вся інформація, із якою формується веб-сторінка, а також її адреса зберігається та отримується із онтології.

Завдяки представленому підходу відбувається спрощення проектування та розробки інтерфейсу користувача для веб-проектів, а в перспективі й для інших додатків (настільних, мобільних та ін.).У порівнянні із класичним методом розробки та проектування інтерфейсу користувача, запропонований

підхід підвищить можливість повторного використання вже розроблених елементів інтерфейсу користувача. А також забезпечить створення бази готових рішень для розробника у вигляді корпоративної пам'яті.

Ключові слова: *база знань, інтелектуальна модель, модель управління пам'яттю, інтелектуальний аналіз даних, ontology-driven development, корпоративна пам'ять.*

Объектом исследования является процесс автоматизированного создания веб-контента, основанный на информации, представленной в онтологическом виде. Одним из самых проблемных мест в веб-разработке является процесс создания интерфейса пользователя. Это связано с тем, что данный процесс является комплексным и требует затраты большего количества времени и средств по сравнению с другими этапами разработки.

В ходе исследования была применена модель разработки программного обеспечения, основанная на разработке онтологии, а затем программного приложения для её обработки. Данный подход называется «Ontology-driven development» (или процесс разработки программного обеспечения (ПО), управляемый онтологией).

Получено интеллектуальную модель для представления элементов веб-ресурсов в виде онтологии, а также программное ядро системы для генерации веб-страниц на основе информации, хранящейся в онтологии. Это связано с необходимостью получения набора конечных элементов интерфейса пользователя (HTML, CSS, JS элементов), из которых формируются веб-страницы.

Налаживание адресации между страницами веб-ресурса имеет ряд особенностей, в частности, для связывания адреса конечной веб-страницы с контроллером, который отвечает за генерацию её содержимого, было предложено соответствующий подход. Этот подход функционирует аналогично так называемому «роутеру», который используется в классических системах (например, JSP для Java). Отличительной особенностью данного подхода является то, что вся информация, из которой формируется веб-страница, а также её адрес хранится и загружается из онтологии.

Благодаря представленному подходу упрощается процесс проектирования и разработки пользовательского интерфейса для веб-проектов, а в перспективе и для других приложений (настольных, мобильных и др.). По сравнению с классическими методами разработки и проектирования интерфейса пользователя, предложенный метод повысит возможность повторного использования уже разработанных элементов интерфейса пользователя. А также обеспечит создание базы готовых решений для разработчика в виде корпоративной памяти.

Ключевые слова: *база знаний, интеллектуальная модель, модель управления памятью, интеллектуальный анализ данных, ontology-driven development, корпоративная память.*

1. Вступ

В наш час, однією із найбільш актуальних завдань є ефективне управління

інформаційними ресурсами. Управління інформаційними ресурсами включає в себе ряд завдань: зберігання великих обсягів інформації, розширення бази знань, редагування існуючої інформації, ефективне представлення інформації користувачеві.

Однією із ключових характеристик у розробці програмних додатків є його собівартість, що декларує конкурентну здатність на ринку розробленого програмного рішення. Дана характеристика напряму залежить від кількості часу, який було витрачено на розробку. На практиці, для скорочення часу на розробку програмного рішення ефективним показав себе метод повторного використання готового протестованого програмного коду. Одним із найбільш витратних, з точки зору часу, є етап проектування інтерфейсу користувача (UI). Запропонований метод дозволяє в перспективі знизити час на розробку програмного додатку завдяки автоматичній генерації елементів UI на основі інформації, що міститься у базі знань.

Таким чином, використання запропонованого метода та моделі знань не тільки скорочує час на розробку програмного забезпечення (ПЗ), а отже, і його вартість, окрім цього, підвищує можливість повторного використання готових рішень у проектуванні UI.

Основним недоліком представленого методу є обмеження його використання системами, що побудовані на основі онтологічних баз знань. Для використання запропонованого методу необхідно окремо, або у складі основного сховища даних зберігати онтологію та налаштовувати взаємодію із нею.

2. Об'єкт дослідження та його технологічний аудит

Об’єктом дослідження є процес створення структурних елементів UI для веб-сторінок. Основна ідея полягає в забезпеченні автоматичної генерації елементів UI на основі інформації, що зберігається в базі знань (БЗ) в онтологічному вигляді. В рамках даної роботи, на прикладі «Порталу Незалежного Оцінювання Якості Вищої Освіти» [1] було розглянуто можливість створення онтологічного опису елементів UI та генерацію на основі цього опису UI для веб-сторінок з роботи [2]. Зокрема, до структурних елементів, що описані в онтологічному вигляді відносяться: шаблони веб-сторінок, блокові та структурні елементи.

Готові веб-сторінки генеруються за допомогою програмного ядра системи, що розроблено на мові програмування Java на основі онтологічного опису, розташованого у базі знань (БЗ) на виділеному сервері. Основні елементи веб-сторінок описані в ієрархічному вигляді.

В рамках розробленої системи, враховуючи особливості побудови онтологій, виділено три основні сутності, на основі яких описано веб-сторінки: «Класи», «Властивості» та «Інстанси» (або кінцеві об’єкти, що є екземплярами класів). Потрібно зауважити, що онтологічний підхід для роботи із даними поділяє «Властивості» на дві групи «Об’єктні Властивості» (Object Property) та «Властивості Типів Даних» (Datatype Properties). Об’єктні властивості поєднують екземпляри класів між собою. В свою чергу, властивості типів даних поєднують екземпляри класів та кінцеві дані. Класи та властивості представлено в ієрархічному вигляді. Діаграму класів представлено на рис. 1.

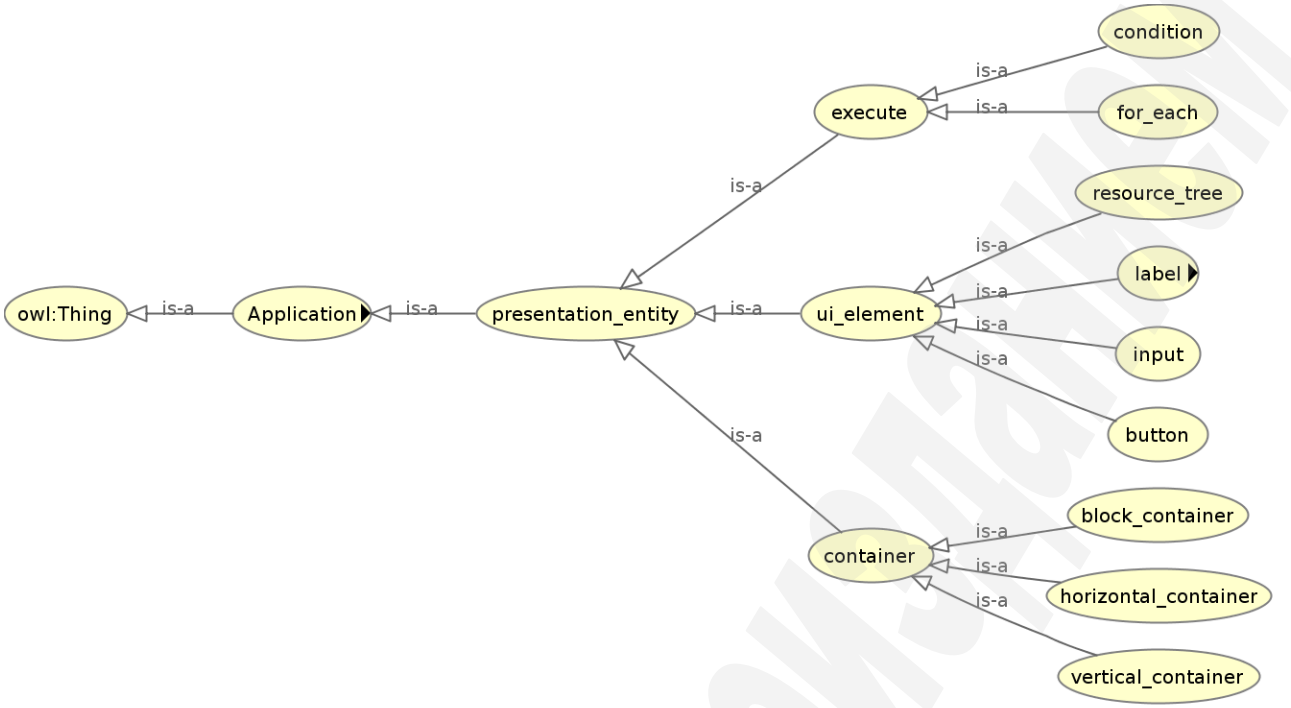

Рис. 1. Діаграма класів

Слід зазначити, що на рис. 1 зображено діаграму розробленої ієрархії класів, що адаптована для розробки веб-сторінок. Дана структура може бути розширена та адаптована для генерації UI для ряду інших технологій, наприклад, додатків Android (інтерфейсу на XML), WPF (XAML-розмітки), Java FX (FXML-розмітки) та програмних онтологічних систем з вбудованим проактивним компонентом [3]. Набір класів регламентує можливі структурні елементи, на основі яких далі буде побудовано онтологічну модель для створення UI.

Онтологічна модель, або онтологічне представлення веб-сторінки, є основою для формування HTML-структури документу. Інтерпретацією онтологічної моделі в кінцевий веб-інтерфейс сторінки займається ядро системи.

Ієрархічна структура представлення елементів UI в якості кореневого елемента має клас «Application».

Клас «presentation entity» наслідується від «application» та поєднує основні класи груп елементів, на основі яких будуватиметься UI: «execute», «ui_element» та «container».

Клас «execute» поєднує класи для створення динамічних елементів UI: цикли, умовні оператори, тощо. Наприклад, на основі класу «for_each» можна сгенерувати елементи для відображення списків на основі добірки даних, що була отримана із БЗ.

Клас «ui_element» поєднує класи для створення основних стандартних елементів UI:

– «resource_tree», що використовується для генерації ієрархічних структур на основі даних із БЗ;

– «label», що є універсальним елементом відображення даних, що були отримані із БЗ. Даний клас використовується для генерації елементів для

відображення заголовків, властивостей ресурсів, тощо. Клас «label» також має дочірній елемент «value_of», що використовується, зокрема всередині циклу «for_each» для доступу до списку отриманих елементів (строк), отриманих в результаті SPARQL-запиту;

– «input» – клас, на основі якого генеруються елементи вводу. Даний клас має властивість «value» для генерації значень елементу за замовчуванням (за аналогією атрибуту «value» в HTML). Окрім цього, «input» має властивість «caption», що дозволяє генерувати відповідний елемент із маскою для вводу (за аналогією атрибуту «placeholder» в HTML) та «name» для доступу до введених даних та їх обробки;

– «button» – клас, на основі якого генеруються елементи кнопки. Даний клас має три атрибути «caption», «value» та «name», які використовуються для генерації атрибута заголовка кнопки, та пар ім'я «name» – значення «value», відповідно, із яких формується інформація, що буде відправлена на сервер;

– «container» – клас, на основі якого генеруються специфічні елементи розмітки сторінки для вертикального «vertical_container» та горизонтального «horizontal_container» розташування елементів. Окрім цього, на основі контейнерів «block_container» генеруються HTML-код для ряду поєднаних в один блок елементів UI. Такими відповідними HTML-елементами можуть виступати , , їх комбінації та ін.

Важливим аспектом є гнучкість та розширюваність запропонованого методу. Так, для того, щоб створити новий, додатковий, наприклад, блочний елемент, немає необхідності вносити структурні зміни в існуючу ієрархію класів та зв'язки між ними, що може ініціювати процес структурних модифікацій програмного ядра системи. Натомість, достатньо, в разі розширення елементної бази UI, додати новий Instance існуючого класу або новий клас та, за необхідності, нові властивості у базу знань. Також, дану систему можна модифікувати для роботи із іншими видами інтерфейсів. При цьому, немає необхідності видаляти існуючу елементну базу; замість цього, нові елементи додаються до вже присутніх у БЗ. Таким чином, в разі зміни предметної області, замість процесів заміни та видалення використовується розширення. Це дає змогу використовувати БЗ однаково як для нової предметної області, так і для вже існуючої; для різних типів додатків або галузей використання.

Недоліками даного методу є підвищена складність первинного проектування та відносно великий об'єм даних, що необхідно створити. Ці фактори значно збільшують собівартість початкової розробки системи. Також є ризик суттєвої втрати продуктивності роботи системи при значному збільшенні об'єму онтології.

3. Мета та задачі дослідження

Мета дослідження – розробка ефективного методу для автоматизованого створення інтерфейсу користувача на основі представлених в онтологічному вигляді, структурних елементів веб-сторінки [4].

Для досягнення поставленої мети необхідно виконати такі задачі:

1. Спроектувати та розробити онтологічний опис структурних елементів інтерфейсу користувача.

2. Розробити програмне ядро системи для генерації структурних елементів веб-сторінок на основі інформації, що міститься в онтології.

3. Розробити метод адресації веб-сторінок всередині веб-ресурсу.

4. Оптимізувати процеси обробки онтології [5, 6].

4. Дослідження існуючих рішень проблеми

Серед основних напрямків вирішення проблеми автоматизованої розробки інтерфейсу користувача, виявлених в світових ресурсах, можна виділити [7], але в даному запропонованому підході не розглянуто можливість автоматизованої генерації інтерфейсу користувача.

В роботах же [8, 9] розглянуто можливість створення інтерфейсу користувача для різних технологій на основі так званого «Model Driven Architecture» підходу, проте не розглянуто можливість створення бази вже створених елементів UI для забезпечення їх повторного використання.

Авторами роботи [10] розглянуто метод автоматичної генерації інтерфейсу користувача, побудований на використанні технологій XML, для JavaFX-додатків, проте не розглянуто можливість адаптування розробленого UI для інших технологій.

У [11] показано можливість створення інтерфейсу користувача для різних платформ, спираючись на «Model Driven Development» підхід, але залишається відкритим питання кінцевої інтеграції вихідного UI в реальні системи.

Альтернативний варіант вирішення проблеми викладений в [12], в якому представлено технологію створення інтерфейсу користувача для інтернету речей (Internet of Things), що не передбачає можливість формування елементів UI для сумісних технологій (напр. GUI для настільних додатків).

Автор роботи [13] розглядає можливість створення людино-машинного інтерфейсу за рахунок розробки спеціальної мови програмування, але залишається питання можливості розширення існуючої бази UI елементів.

В [14, 15] представлено технології формування інтерфейсу користувача, в специфічних технологіях для веб та мобільних технологій відповідно. Проте, залишається не розкритим процес інтеграції створеного UI в існуючі системи та можливість його адаптації для інших технологій.

А ось робота [16] присвячена проблемі створення інтерфейсу користувача для основних видів ПЗ: веб-додатків, мобільних пристроїв. Але в цій роботі не передбачено можливість створення UI для настільних додатків, також, недостатньо розкрито процес зберігання програмного коду для вже розроблених елементів UI.

Таким чином, результати літературного аналізу дозволяють зробити висновок про те, що представлений підхід поєднує сильні сторони аналогів, враховує їх недоліки та розширює сферу застосування перерахованих методів.

5. Методи досліджень

За основу, в рамках даного дослідження, було обрано так званий ODD (Ontology-Driven Development – метод розробки програмних додатків, що

передбачає спочатку створення онтології, а потім програмного коду) [17]. Цей підхід полягає в проектуванні та створенні онтології на першому етапі розробки та написанні програмного коду на другому етапі. В ході дослідження систем, що передбачають адаптацію до зміни предметної галузі, було виявлено переваги використання даного підходу [6].

Для проектування та розробки програмного ядра було використано стек технологій, побудованих на Java [18], основним із яких є Apache Jena [19]. Для доступу до інформації, що зберігається в БЗ було використано SPARQL мову запитів [20]. Для зберігання та обробки БЗ було використано універсальний сервер Virtuoso [21]. Окрім основного призначення, Virtuoso також забезпечує додаткову оптимізацію процесів взаємодії із БЗ за рахунок створення об'єктної моделі на основі завантаженої онтології. Дана модель також використовується Jena, що спрощує процес доступу до інформації, що зберігається у БЗ.

6. Результати досліджень

В результаті досліджень було створено метод генерації веб-контенту на основі онтології, було спроектовано та розроблено онтологічну модель для опису структурних елементів UI для веб-сторінок. Приклад сторінки, для якої було створено онтологічний опис, представлено на рис. 2.

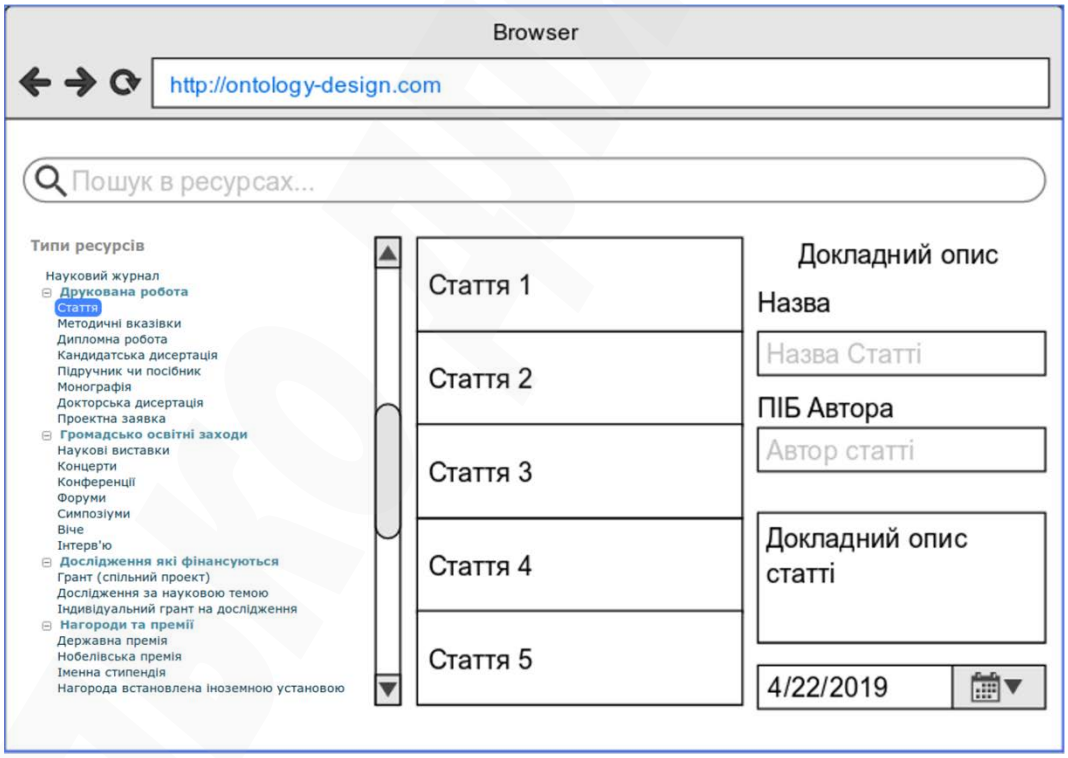

Рис. 2. Приклад веб-сторінки

Представлена на рис. 2 веб-сторінка складається із таких елементів: поле для вводу даних, горизонтальний блок, в якому знаходиться зліва-направо: ієрархічна структура ресурсів, розташованих в БЗ; перелік обраних ресурсів; докладний опис ресурсу. Онтологічний опис веб-ресурсу представлений на рис. 3.

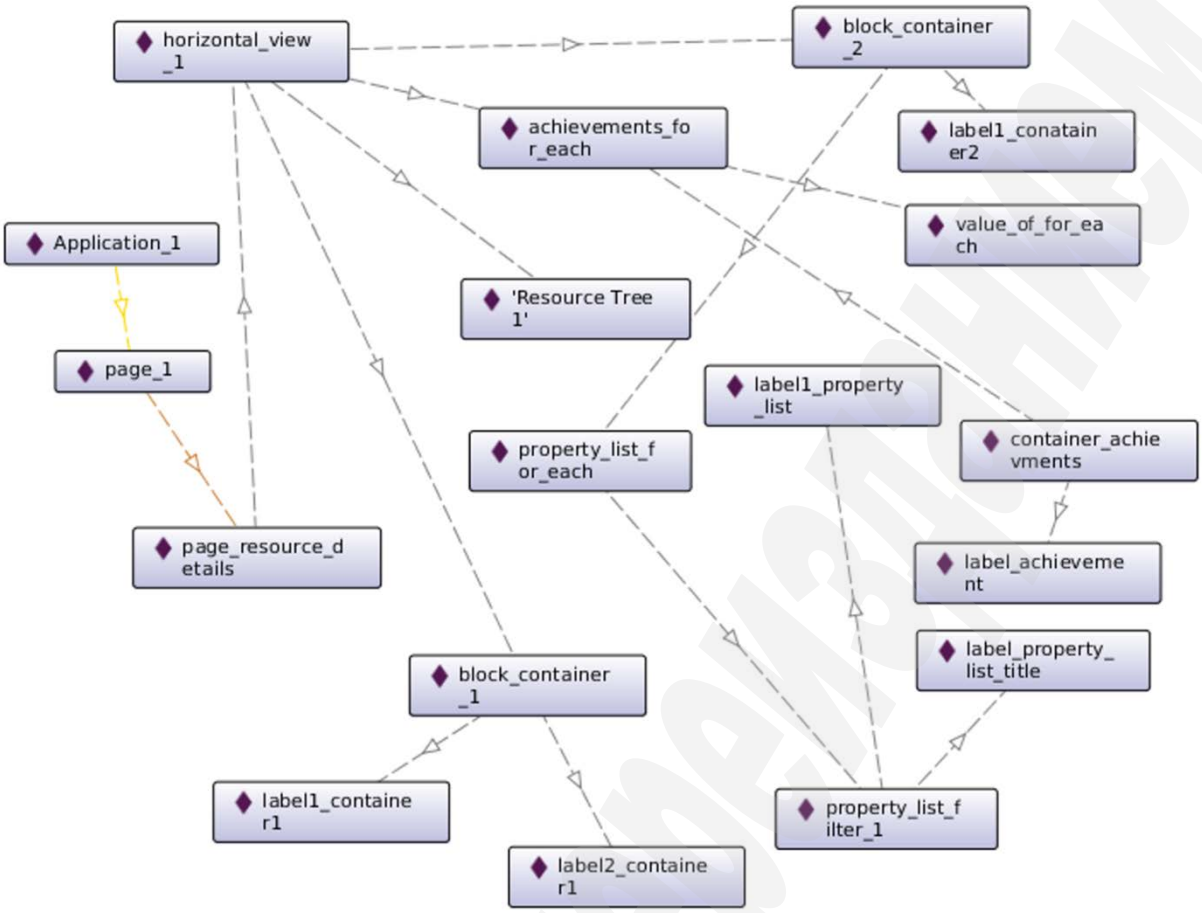

Рис. 3. Онтологічний опис веб-ресурсу

На рис. 3 зображено онтологічний опис кінцевого веб-ресурсу, представлений в об'єктному вигляді. Базовою одиницею, що використовується для опису виступає об'єкт, що в рамках онтології має визначення «Instance».

«Instance» – це об'єкт, побудований на основі конкретного класу, що пов'язаний об'єктними властивостями або властивостями типів даних із іншими об'єктами. Instance також називають ресурсом. Instance можуть містити конкретні дані або інші Instance. Для того, щоб розмістити в Instance об'єкти використовуються об'єктні властивості; для того, щоб в Instance додати дані, використовують властивості типів даних.

Слід зазначити, що всі дані в онтології є типізованими. Складні типи даних можуть бути створені на основі вже існуючих в онтології. Так, в процесі розробки онтології було створено додаткові типи даних: «elementType», «sparqlCode», «updateFreaquency».

Запропонована структура елементів UI побудована у вигляді ієрархії, яка має кореневий елемент (root), сестринські (siblings) та дочірні (child) елементи. Кореневим елементом виступає «Application_1», який поєднує всі сторінки веб-ресурсу.

Для структури окремої веб-сторінки, яка представлена на рис. 3, кореневим елементом є «page_1». Даний об'єкт містить адресу, за якою розміщена сторінка, а також ім'я контролера, який із нею пов'язаний, що використовується ядром системи для генерації веб-контенту за онтологічною моделлю. Дана інформація представлена в онтології за допомогою властивостей типів даних.

«page_resource_details» є об’єктом класу «page_view» та включає в себе елементи, із яких побудована сторінка. В даному випадку, «page_resource_details» включає єдиний елемент «horizontal_view_1», що в контексті онтології означає, що «page_resource_details» пов’язаний через об’єктну властивість «element» із «horizontal_view_1».

«horizontal_view_1» є об’єктом класу «horizontal_container», що успадковується від класу «container» та визначає розташування елементів UI на сторінці. «horizontal_view_1» представляє побудову сторінки у вигляді структурних елементів, що розташовані горизонтально відносно один-одного. Даний об’єкт пов’язаний через об’єктну властивість «element» із «block_container_1», «Resource Tree 1», «achievements_for_each» та «block_container_2».

«Resource Tree 1» – об’єкт класу «resource_tree», який наслідується від класу «ui_element». Даний елемент використовується для опису ієрархічної структури даних, що буде побудовано на сторінці.

«block_container_1» та «block_container_2» є об’єктами класу «block_container», що наслідується від класу «container» та представляють опис блокових елементів веб-сторінки (наприклад, «div»).

Елементи-контейнери (тобто об’єкти класів, що успадковуються від класу «container» або його класів-спадкоємців) можуть включати в себе прості елементи із яких буде згенеровано UI – це об’єкти класів: «label», «button», «input», а також об’єкти, що використовуються для динамічного створення елементів UI: «for_each», «condition». Класи об’єктів першої групи елементів є спадкоємцями «ui_element». Класи об’єктів другої групи успадковується від класу «execute».

Об’єкти класу «execute» використовуються для динамічної генерації елементів UI. На основі даних, що мітяться в цих елементах, програмне ядро може згенерувати елементи веб-сторінки для відображення даних із бази знань: «achievements», «property_list». Дані елементи відповідають структурним елементам HTML «select» та «option» відповідно.

Повний перелік елементів, що міститься в онтології із указаними зв’язками між ними (Object Property, Datatype Property, Datatype) наведено на рис. 4.

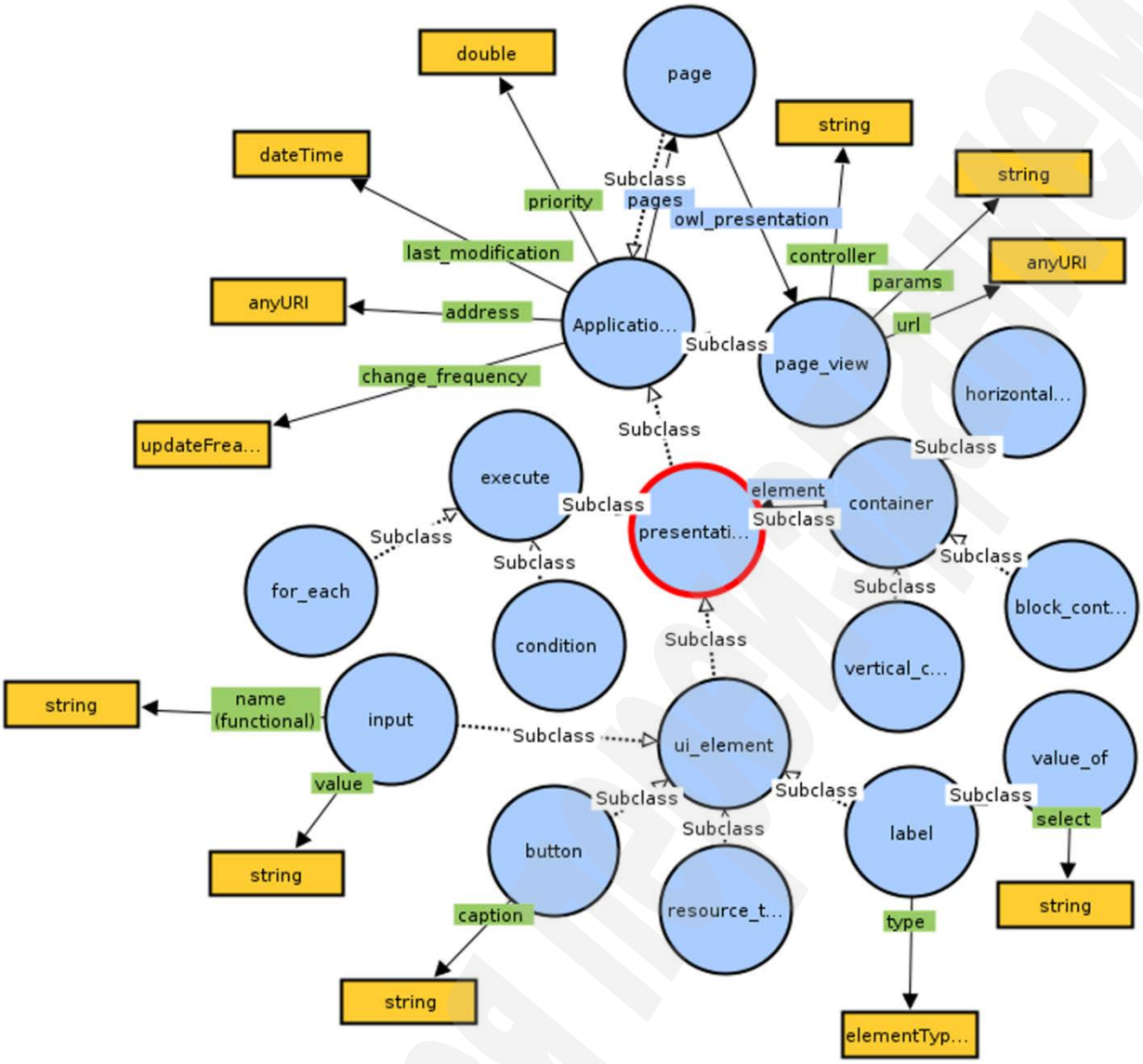

Рис. 4. Схематичне зображення онтології із властивостями

На рис. 4 представлено структуру онтології із зазначеними властивостями, що зв'язують об'єкти класів. Окрім цього на рис. 4 наведено типи даних, на які вказують властивості типів даних.

З точки зору економічної вигідності, використання даного методу знизить у перспективі витрати часу на проектування та розробку UI та, як наслідок, знизить собівартість розробки. Окрім цього, використання даного підходу, в перспективі, підвищить шанси уніфікувати UI для веб-ресурсів та полегшить мігрування готового UI для інших технологій, таких як «WPF», «fxml» та ін.

7. SWOT-аналіз результатів досліджень

Strengths. Використання даного метода, в перспективі, дозволить знизити витрати на проектування та розробку UI для проектів різного роду: веб-ресурсів, настільних додатків, додатків для мобільних пристроїв та ін. Це можливо за рахунок уніфікування опису вже створених елементів UI, формування із них бази, що може розширюватися або змінюватися; при цьому мінімізується вплив змін на вже існуючі елементи у БЗ

У порівнянні із аналогами даної системи, скорочується собівартість розробки та збільшується продуктивність розробки UI в цілому.

Weaknesses. До слабких сторін даного методу відносяться необхідні первинні затрати часу на проектування елементів, на основі яких буде побудовано UI. Окрім цього, використання даного підходу збільшить складність розробки та збільшать її первинну трудомісткість.

Opportunities. До перспектив розвитку даного методу відноситься розширення вже існуючого набору описів елементів UI, спрощення проектування онтології та додавання в неї даних, шляхом розробки специфічного програмного додатку автоматизованої генерації онтологічного опису елементів за його описом на природній мові спілкування. Використання запропонованого підходу на підприємстві дозволить кінцевим розробникам створювати спільний простір у вигляді корпоративної пам'яті для зберігання створених описів елементів UI та практики їх впровадження в готові рішення.

Threats. До загроз основних загроз, пов'язаних із застосуванням даного методу відноситься ризик використання методів, які більш легкі для впровадження на підприємстві. Окрім цього, для підтримки проектування бази знань, в якій буде міститися опис елементів UI, потрібен фахівець, який має кваліфікацію в застосуванні онтологій. У зв'язку із цим може виникнути необхідність найняти додатково співробітника на підприємство, або вкладати кошти у розвиток спеціалістів, які вже на підприємстві є, для здобуття необхідних знань та кваліфікації в даній області.

8. Висновки

1. Спроектовано та розроблено онтологічний опис структурних елементів інтерфейсу користувача. В результаті отримано узгоджену онтологію для зберігання елементів розробленого інтерфейсу користувача. Розроблена онтологія складається початково із 17 базових класів, 15 властивостей та 18 об'єктів. А також передбачає можливість розширення для інтегрування елементів UI суміжних технологій (мобільних рішень, настільних додатків) та доповнення елементами UI для створення веб-додатків.

2. Розширене програмне ядро системи [1] для забезпечення можливості генерації структурних елементів веб-сторінок на основі інформації, що міститься в онтології. В результаті було розроблено додаткові класи та методи програмного ядра системи, що були інтегровані в існуючу архітектуру системи.

3. Розроблено метод адресації веб-сторінок всередині веб-ресурсу. В результаті цього основна онтологія була розширена для забезпечення зберігання внутрішніх посилань. Також було розширене програмне ядро системи ієрархією класів та методами для організації адресації між сторінками всередині веб-ресурсу.

4. Після впровадження необхідних модифікацій в ядро системи та створення а також інтегрування додаткової онтології було виявлено втрату продуктивності системи на 5–10 %. Для відновлення швидкодії системи впроваджено комплекс методів для оптимізації роботи, які описані в [22]. В результаті було отримано приріст продуктивності системи в середньому на 10–15 %.

Література

1. *TRUST Portal*. Available at: http://portal.dovira.eu/ Last accessed: 17.10.2019

2. Terziyan, V., Golovianko, M., Shevchenko, O. (2014). Semantic Portal as a Tool for Structural Reform of the Ukrainian Educational System. *Information Technology for Development, 21 (3),* 381–402. doi: http://doi.org/10.1080/02681102.2014.899955

3. Terziyan, V., Shevchenko, O., Golovianko, M. (2014). An introduction to knowledge computing. *Eastern-European Journal of Enterprise Technologies, 1 (2 (67)),* 27–40. doi: http://doi.org/10.15587/1729-4061.2014.21830

4. Shevchenko, A. Iu., Shevchenko, E. L. (2011). How to bring artificial intelligence into the Clouds. *Eastern-European Journal of Enterprise Technologies, 3 (12 (51)),* 66–70. Available at: http://journals.uran.ua/eejet/article/view/2472

5. Bibichkov, I. E., Sokol, V. V., Shevchenko, A. Iu. (2014). Optimizing the performance of ontological knowledge bases built on the basis of «VIRTUOSO». *Eastern-European Journal of Enterprise Technologies, 5 (2 (71)),* 4–8. doi: http://doi.org/10.15587/1729-4061.2014.28553

6. Shevchenko, A. Y., Shevchenko, E. L. (2012). Modern ontological database management systems comparison. *Visnik SEVNTU, 131,* 82–86.

7. Offenhartz, J. K., Dana, D. (2017). *Dynamic generated WEB UI for configuration*. published: 17.04.17.

8. Bernaschina, C., Comai, S., Fraternali, P. (2017). Online Model Editing, Simulation and Code Generation for Web and Mobile Applications. *2017 IEEE/ACM 9th International Workshop on Modelling in Software Engineering (MiSE),* 33–39. doi: http://doi.org/10.1109/mise.2017.1

9. Roubi, S., Erramdani, M., Mbarki., S. (2016). A Model Driven Approach for generating Graphical User Interface for MVC Rich Internet Application. *Computer and Information Science, 9 (2),* 91. doi: http://doi.org/10.5539/cis.v9n2p91

10. Yongzhi, Y., Peng, Z., Yun, X. (2017). Automatic User Interface Generating for Simple Interaction in Pervasive Computing. *22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2,* 3–8. doi: http://doi.org/10.1109/cse-euc.2017.187

11. Bouraoui, A., Gharbi, I. (2019). Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. *Science of Computer Programming, 172,* 63–101. doi: http://doi.org/10.1016/j.scico.2018.11.002

12. Johnsson, B. A., Magnusson, B. (2017). Towards end-user development of graphical user interfaces for internet of things. *Future Generation Computer Systems.* doi: http://doi.org/10.1016/j.future.2017.09.068

13. Gaouar, L., Benamar, A., Le Goaer, O., Biennier, F. (2018). HCIDL: Human-computer interface description language for multi-target, multimodal, plastic user interfaces. *Future Computing and Informatics Journal, 3 (1),* 110–130. doi: http://doi.org/10.1016/j.fcij.2018.02.001

14. Taivalsaari, A., Mikkonen, T., Systä, K., Pautasso, C. (2018). Web User Interface Implementation Technologies: An Underview. *Proceedings of the 14th International Conference on Web Information Systems and Technologies, 1,* 127–136.

doi: http://doi.org/10.5220/0006885401270136

15. Yannes, Z., Tyson, G. (2019). Amniote: A User Space Interface to the Android Runtime. *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, 1,* 59–67. doi: http://doi.org/10.5220/0007715400590067

16. Engel, J., Märtin, C., Forbrig, P.; Kurosu, M. (Ed.) (2017). Practical Aspects of Pattern-Supported Model-Driven User Interface Generation. Human-Computer Interaction. User Interface Design, *Development and Multimodality. Cham: Springer International Publishing*, 397–414. doi: http://doi.org/10.1007/978-3-319-58071-5_30

17. Gharbi, G., Ben Alaya, M., Diop, C., Exposito, E. (2012). AODA: An autonomic and ontology-driven architecture for service-oriented and event-driven systems. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WETICE, 3,* 72–77. doi: http://doi.org/10.1109/wetice.2012.84

18. *The Java™ Tutorials*. Available at: https://docs.oracle.com/javase/tutorial/ Last accessed: 24.03.2019

19. *Apache Jena – Jena tutorials.* Available at: https://jena.apache.org/tutorials/index.html Last accessed: 24.03.2019

20. Zhang, Y. (2015). Research on Efficient SPARQL Query Processing for RDF Data. *Proceedings of the 2015 2nd International Workshop on Materials Engineering and Computer Sciences*, 476–482. doi: http://doi.org/10.2991/iwmecs-15.2015.94

21. *OpenLink Virtuoso Universal Server : Documentation* (2007). Available at: http://docs.openlinksw.com/virtuoso/

22. Bibichkov, I., Sokol, V., Shevchenko, O. (2017). Ontological knowledge bases productivity optimization through the use of reasoner combination. *Eastern-European Journal of Enterprise Technologies, 5 (2 (89)),* 49–54. doi: http://doi.org/10.15587/1729-4061.2017.112347

The object of research is the process of automated creation of web content based on information presented in an ontological form. One of the most problematic places in web development is the process of creating a user interface. This is due to the fact that this process is complex and requires more time and money than other development stages.

During the study, there was applied a software development model, which is, based on the development of an ontology, and then a software application for its processing. This approach is called «Ontology-driven development» (or the software development process controlled by the ontology).

An intellectual model is obtained for representing the elements of web resources, which is represented in the form of an ontology, as well as the software core of the system for generating web pages, based on information stored in the ontology. This is due to the process, which is need to obtain a set of finite elements of the user interface (HTML, CSS, JS elements) from which web pages are formed.

Setting addressing between pages of a web resource has a number of features. In particular, an appropriate approach was proposed for linking the address of the final web page with the controller, which is responsible for generating its content. These functions are similarly to the so-called «router», which is used in classical web-based systems (e. g. JSP for Java). A distinctive feature of this approach is in the set of

information from which the web page is formed. This information, as well as its address is stored and loaded from the ontology.

Due to the presented approach, the process of designing and developing of the user interface is simplified in comparison with classical ones. This approach is effective for web projects and, in the perspective, for other applications (desktop, mobile, etc.). Also, the proposed method will increase the possibility of reusing already developed elements of the user interface, as well as ensure the creation of a base of ready-made solutions for the developers in the form of corporate memory.

Keywords: *knowledge base, intellectual model, memory management model, data mining, ontology-driven development, corporate memory.*