УДК 681.518:658.512

УНІФІКОВАНІ ПРОГРАМНІ СЕРВІСИ ТА ВІЗУАЛЬНІ ІНТЕРФЕЙСИ В ІНТРАНЕТ-СИСТЕМАХ УПРАВЛІННЯ ТЕХНОЛОГІЧНИМИ ПРОЦЕСАМИ

М.В. ТКАЧУК, В.А. ШЕХОВЦОВ

Розглянуто деякі проблеми побудови інтранет-базованих інформаційно-управляючих систем і представлена їх типова функціональна структура. На підставі її аналізу сформульовано вимоги до однієї з найбільш важливих функцій таких систем — візуалізації телеметричних даних — і запропоновано уніфіковану архітектуру відповідних програмних сервісів, яку реалізують альтернативні варіанти використання сучасних web-технологій. Наведено приклади розроблених програмних рішень і візуальних інтерфейсів користувача.

АКТУАЛЬНІСТЬ ТЕМИ, ДЕЯКІ ПРОБЛЕМИ ТА МЕТА ДОСЛІДЖЕННЯ

Незважаючи на те, що за останні роки значно розвинулись сучасні методи та технології розробки та реінжинірингу програмного забезпечення, створення складних, тобто багаторівневих та розподілених інформаційно-управляючих систем (ІУС), що широко застосовуються практично в усіх галузях промисловості та економіки, залишається вельми актуальною проблемою. Це обумовлено великою розмірністю, комплексною функціональністю та високою вартістю розробки нових ІУС, а також необхідністю збереження матеріально-технічних ресурсів, вже інвестованих в діючі, або так званих успадкованих систем (*legacy system*). В умовах позитивних змін в промислових галузях економіки саме питання зростання ефективності автоматизації технологічних процесів (ТП) «*...повинні стати найбільш пріоритетним напрямком інформатизації в Україні*» [1].

Одним із перспективних напрямків розробок у сфері ІУС АСУ ТП є застосування концепції та архітектури так званих SCADA (Supervisory Control and Data Acquisition)-систем [2, 3], які забезпечують збір та обробку даних у реальному масштабі часу за допомогою різноманітних програмованих логічних контролерів (*programmable logical controller-PLC*). Функції типової SCADA-системи визначаються двома великими групами сервісів: обробка телеметричних даних і безпосереднє управління параметрами ТП. Вони, у свою чергу, складаються з окремих класів задач (рис. 1) у вигляді відповідної діаграми в нотації UML (Unified Modeling Language) [4]).

Здебільша SCADA-системи будуються за традиційною технологією обробки даних «клієнт—сервер», і типовим прикладом такого підходу є практично єдина зараз на ринку SCADA-систем в Україні вітчизняна розробка – система «Контур-2» [5]. Вона, безперечно, має деякі позитивні риси, але, на наш погляд, саме її дворівнева архітектура обробки даних із застосуванням технології ActiveX для візуального відображення даних ускладнює застосування цієї системи у випадку необхідності побудови багаторівневих розподілених ІУС в АСУ ТП.

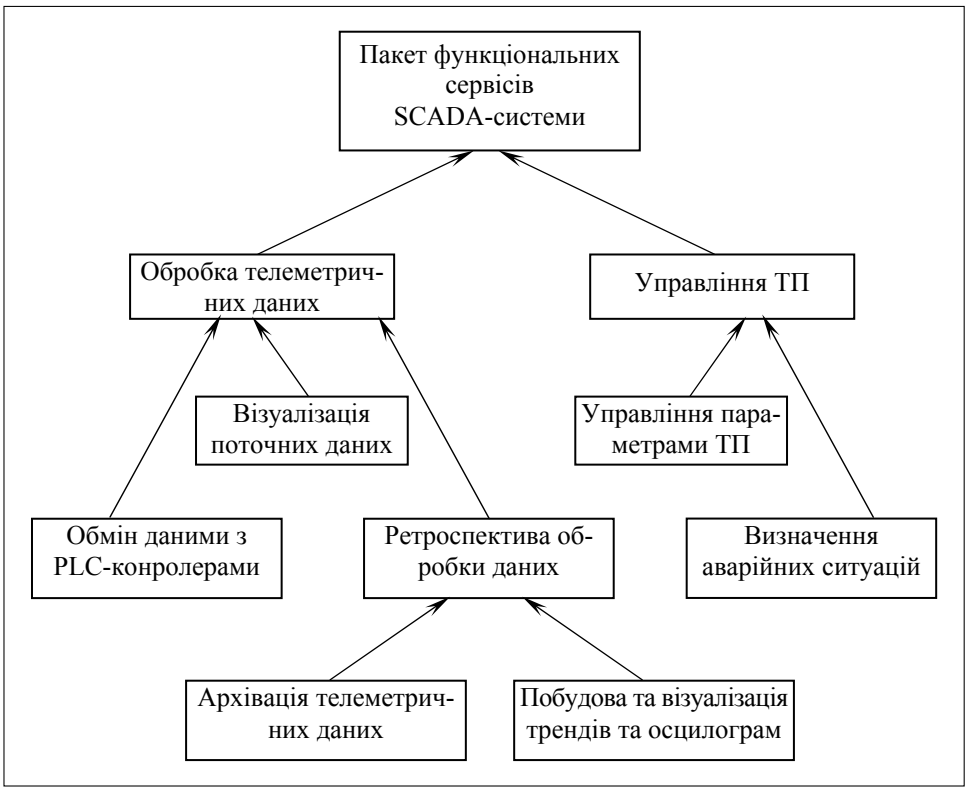

Рис. 1. Структура функціональних сервісів типової SCADA-системи

Саме тому зараз спостерігається тенденція до переходу SCADA-систем на платформу Web-технологій [3, 6]. У попередніх публікаціях ми запропонували один із можливих підходів до побудови Web-базованої SCADA як інтранет-системи з власною DOM (Document Object Model)-моделлю для представлення вибору формату та обробки даних. Найбільш критичним при цьому з точки зору продуктивності отриманого коду є проблема ефективної реалізації сервісів візуалізації телеметричних даних відповідного ТП, які повинні накопичуватися та оброблятися в SCADA-системі у реальному масштабі часу. Далі ми розглянемо декілька альтернативних підходів до вирішення цієї проблеми та запропонуємо уніфіковану архітектуру відповідних програмних сервісів. Ці рішення були розроблені та експериментально перевірені в кількох проектах по створенню SCADA-систем на об’єктах нафтогазовидобування в Харківській та Дніпропетровській областях [7–9].

ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ВІЗУАЛІЗАЦІЇ ДАНИХ У WEB-SCADA-СИСТЕМАХ

Візуалізація телеметричних даних, які накопичуються та обробляються у Web-SCADA-системі, повинна відповідати, на наш погляд, таким функціональним вимогам:

1. Користувачеві системи треба надати можливість працювати із стандартним Web-браузером (*browser*) для моніторингу відповідного ТП у віддаленому режимі (remote mode). Оскільки для такого доступу у реальних системах можуть використовуватись повільні та ненадійні телефонні лінії, між клієнтом та сервером можуть пересилатись лише відносно невеликі обсяги даних.

2. Візуалізація повинна надавати користувачеві чітке відображення поточного стану процесу, що контролюється SCADA-системою. Це представлення має певний набір *візуальних технологічних схем* (ВТС) – двомірних відображень фізичних блоків ТП. Наприклад, візуалізація підсистеми управління ТП обробки нафти може бути реалізована за допомогою таких ВТС, як діаграма блоку теплообмінника, діаграма блоку збереження та перекачування нафти і т. ін. Кожна з таких схем містить набір базових візуальних елементів, що відповідають фізичним об'єктам блоків (резервуарам, насосам, кранам), та візуальне відображення трубних з'єднань. Приклад однієї з конкретних ВТС, які були розроблені у наших проектах, наведено на рис. 2.

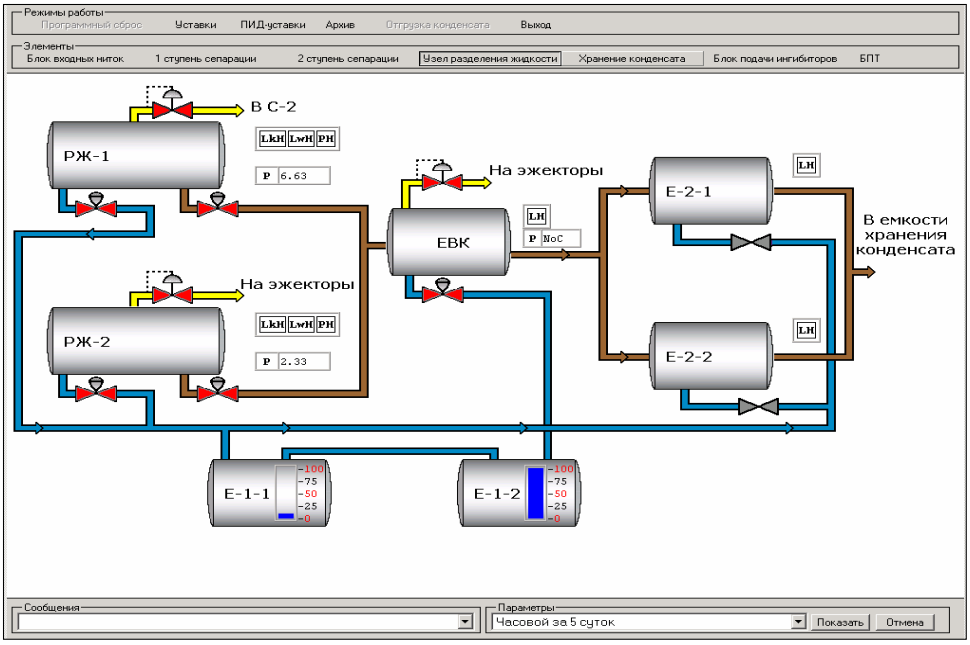

Рис. 2. Приклад візуальної технологічної схеми, побудованої за HTML-технологією

3. На кожній із схем необхідно відображати поточні значення дискретних та аналогових параметрів, отриманих з відповідного PLC-протоколу. Дискретні параметри представляються зміною кольору або аніма-

цією елементів ВТС. Аналогові — у числовому вигляді або індикаторами, які змінюють свої розміри та кольори.

4. Візуальне представлення поточного стану ТП повинно надаватися у режимі реального часу, коли інтервал між запитаннями на отримання нових даних дорівнює близько 400 мс.

5. Необхідно реалізувати також візуалізацію архівних значень параметрів. Інформація з архіву повинна відображатися як лінійна діаграма, де час відкладається по осі абсцис, а нормалізовані значення параметрів — по осі ординат. Треба передбачити можливість відображати декілька (як правило, до п'яти) параметрів на одній діаграмі.

На практиці існують різні режими візуалізації архіву, залежно від деяких параметрів: а) *загального інтервалу часу* для запрошених даних; б) *часового інтервалу візуалізації* (інтервалу часу, який повинен бути відображений на одній діаграмі); в) *кроку візуалізації* (інтервалу між двома точками на діаграмі).

Наведемо приклади режимів візуалізації архіву: «Дані за 2 роки (повний діапазон часу), із відображенням за 48 годин (часовий інтервал візуалізації), погодинно (крок візуалізації)», «Дані за 48 годин, із відображенням за годину, щохвилинне». Користувач повинен мати можливість негайно бачити точні цифрові значення усіх параметрів, що відображуються, для будь-якої точки на діаграмі.

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІЗУАЛІЗАЦІЇ ДАНИХ

Відповідно до вищезгаданих вимог, нами було розроблено архітектуру *розвиненої візуалізації телеметричних даних* (РВТД) як інтегровану частину Web-SCADA-системи (рис. 3). Необхідно підкреслити, що всі ці підходи

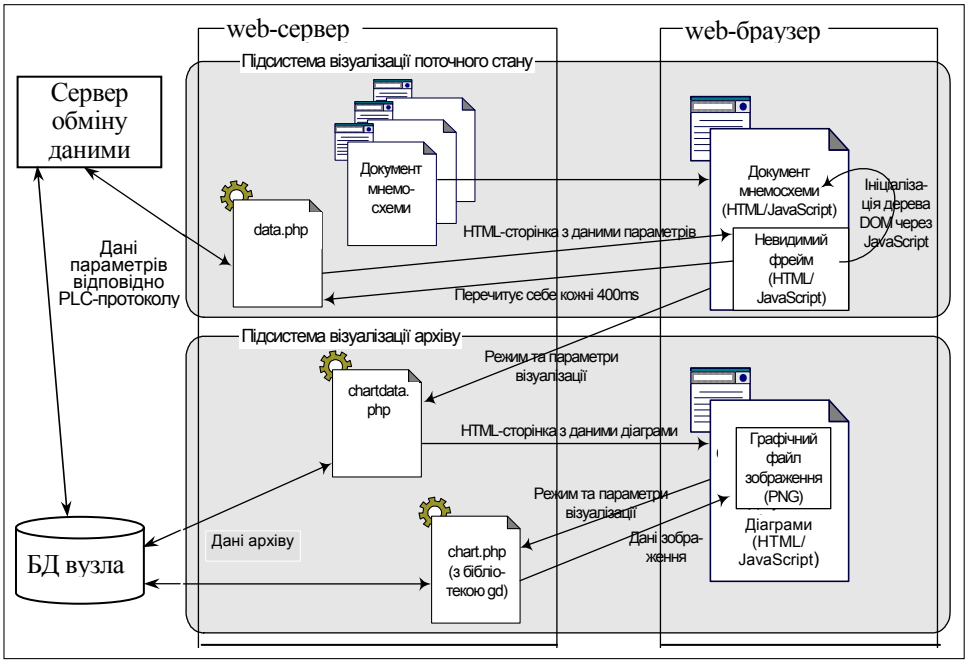

Рис. 3. Архітектура підсистеми РВТД і її взаємодія з іншими компонентами типової Web-SCADA-системи

базуються на Web-орієнтованій клієнт-серверній архітектурі, яка складається з Web-сервера, що динамічно генерує HTML-документи з використанням технології PHP [10] на основі інформації, отриманої із сервера обміну даними (СОД) [7] через COM-інтерфейс (або прямо з бази даних системи) та Web-браузера, що додатково обробляє ці дані. Кожній ВТС відповідає HTML-документ, а базовим візуальним елементам — набори тегів HTML у рамках цього документа.

Система РВТД функціонально складається з двох підсистем.

1. *Підсистема візуалізації поточних даних* (ПВПД) технологічних процесів, яка відображає поточні значення параметрів на ВТС у реальному масштабі часу.

2. *Підсистема візуалізації архіву даних* (ПВАД), яка відповідає за відображення архівних значень.

Нижче більш детально розглянемо можливі програмні рішення, що були вивчені та експериментально випробувані нами для розробки системи РВТД.

ІМПЛЕМЕНТАЦІЯ ПВПД

Спочатку розглянемо можливі альтернативні шляхи реалізації ПВПД.

Перший підхід — це генерація всіх даних для візуалізації на Web-сервері без додаткової їх обробки. Такий підхід ставить мінімальні вимоги до клієнта, бо можна використовувати будь-яку програму-браузер і тим самим підвищувати мобільність системи. Але відбувається вона ціною погіршення продуктивності та високого завантаження каналу зв'язку. Насправді кожна зміна відображення потребує нової генерації документа схеми.

Інший підхід полягає у використанні Java-аплетів *(applet*), які отримують дані безпосередньо із сервера, встановлюючи окремі з'єднання з ним. До переваг цього підходу відносяться висока гнучкість, мобільність та відповідно висока продуктивність після початкового завантаження (кожен аплет може отримувати тільки потрібні йому дані). У цьому підході є і недоліки: високі вимоги до клієнта, довге початкове завантаження, обмежена та ненадійна підтримка в стандартних браузерах.

Третій підхід (саме його обрано для реалізації ПВПД) – використання динамічного HTML [10]. Підкреслимо основні принципи.

1. Встановлюється зв'язок СОД з PHP-базованою сторінкою (*data.php*), яка завантажується у невидимий HTML-фрейм, що автоматично перечитує себе кожні 400 мс. Значення параметрів, які були отримані з сервера, зберігаються у прихованому елементі форми на цій сторінці.

2. Після завантаження сторінки *data.php* починає роботу код на мові JavaScript. Він читає дані із прихованого елемента, робить їх синтаксичний розбір та згідно із їх значеннями корегує відповідні атрибути елементів та текстових вузлів DOM-дерева поточної сторінки схеми. Атрибути відповідають за колір, розмір індикаторів та анімацію, текстові вузли відображають числові значення аналогових параметрів. Фрагмент цього коду наведено на рис. 4.

```
function Load()     // автоматично викликається під час завантаження сторінки
{
      var paramValues = parent.values;                    // дані з прихованого елемента
      var paramElements = document.all("parameter"); // усі елементи з id=='parameter'
      for (i = 0; i < paramElements.length; ++)
      {
            paramId = paramElements[i].paramId;    // визначення коду елемента
            paramValue = paramValues[parseInt(paramId) - 1];
            // відображення параметра
            parameter_change(paramElements[i],
                              Math.round(parseFloat(paramValue)*100)/100);
     }
}
// відображення параметра
function parameter_change(parameter, mval)
{
      // зміна числового значення параметра
      parameter.innerHTML = mval;
      // зміна стилю елемента індикатора
      var indicator = document.getElementById(parameter.indicator);
      var barmax = indicator.height - 15;
      indicator.getElementById("bar").style.height =
                              Math.round(barmax*mval/100) + "px";
}
```

Рис. 4. Фрагмент програмного коду візуалізації

Такий підхід має декілька переваг.

1. Серверний код є дуже простим.

2. Обсяг обміну даними між клієнтом та сервером мінімальний (точні цифри залежать від PLC-протоколу, але навряд чи будуть більше 10 Кбайт за один обмін).

3. Навантаження на клієнта не дуже високе (код на JavaScript теж дуже простий та виконується швидко).

4. Код може бути перенесений на будь-який браузер, який підтримує стандарт W3C DOM (наприклад, MS Internet Explorer версії 5.0, 6.0 або Netscape 6/Mozilla).

ІМПЛЕМЕНТАЦІЯ ПВАД

Візуалізація даних архіву відрізняється від візуалізації даних поточного стану декількома важливими характеристиками.

1. Візуалізація даних поточного стану на клієнті може бути реалізована досить просто (дії відомі заздалегідь, не потрібні складні розрахунки, вимоги до пам’яті невеликі). З іншого боку, візуалізація даних з архіву потребує побудови відповідних діаграм, що, у свою чергу, потребує виконання складних чисельних розрахунків (наприклад, під час масштабування), які виконуються з масивом даних у 3000 елементів (усі точки часового інтервалу

візуалізації для усіх параметрів). Можливості мови JavaScript не дуже добре підходять до складних розрахунків над такими масивами.

2. Візуалізація даних поточного стану є задачею реального часу, яка повинна виконуватися у визначений часовий інтервал (у нашому випадку 400 мс). Візуалізація архіву, з іншого боку, є інтерактивною операцією, яка виконується тільки на вимогу користувача. Вона повинна відповідати звичайним вимогам до часу реакції (наприклад, цей час не повинен перевищувати 3–5 с), але вони не є вимогами реального часу.

На наш погляд, існує декілька підходів до реалізації ПВАД: а) використання засобів динамічного HTML для ручного відображення діаграми на клієнті (як підкреслено вище, це рішення не масштабується через високе завантаження клієнта); б) застосування готового ActiveX-компонента, такого, наприклад, як MSChart (цей підхід може також значною мірою обмежити мобільність та гнучкість отриманого рішення); в) створення діаграми на Web-сервері як графічного файла та передача цього файла клієнтові для відображення (саме цей підхід і було обрано для реалізації).

У складі ПВАД можна виділити дві основні частини.

1. Програмний код, який відповідає за відображення документа діаграми архіву (*chartdata.php*). Цей документ є HTML-документом, що містить графічне зображення діаграми.

2. Код створення зображення – частина, що відповідає за створення безпосередньо зображення (*chart.php*).

Для того щоб почати роботу з ПВАД, користувачеві потрібно відмітити на ВТС необхідні параметри та вибрати режим візуалізації. Після цього всі необхідні дані передаються до коду *chartdata.php*. Код робить запитання до ІБДВ, отримує дані діаграми для часового інтервалу візуалізації та відображає документ діаграми архіву у окремому вікні. Документ містить код для масиву JavaScript, заповнений даними поточного інтервалу візуалізації (індексованого за часом), пустий шаблон для таблиці візуалізації параметрів та посилання на код створення зображення усередині тега <IMG>. URL для цього посилання містить режим візуалізації та список параметрів. Код *chart.php* спирається на графічну бібліотеку *gd* та на пакет *PHPlot* для створення графічного файла у форматі PNG, який містить діаграму. Дані діаграми отримуються також як відповідь на запити до ІБДВ з використанням інформації, переданої через URL. Після завантаження сторінки та зображення діаграми користувач може пересувати покажчик миші над зображенням. У цьому випадку поточна *X*-координата миші отримується JavaScript-кодом цієї сторінки, перетворюється у значення часу відповідно до діючого масштабу. Після цього елементи масиву, які відповідають цьому моменту часу, відображаються у таблиці візуалізації параметрів під діаграмою (рис. 5).

Цей підхід має такі переваги:

1. Програмний код є простим, користується існуючими бібліотеками (пакети *gd* та *PHPlot*).

2. Між клієнтом та сервером пересилається не дуже багато даних, бо графічний файл компактний (у ньому використовуються не більш, ніж 7 кольорів).

3. Навантаження на клієнта мінімальне. За всю складну обробку відповідає сервер.

Недоліком такого підходу є те, що ми насправді виконуємо два запити для отримання тих самих даних: із *chartdata.php* та із *chart.php*. Можна спробувати передати усю інформацію з *chartdata.php* через URL, але такий підхід важко реалізувати ефективно. Практичне тестування показало, що зниження ефективності від виконання двох однакових запитів замість одного фактично майже непомітне через те, що сучасні СУБД організують хешування запитів.

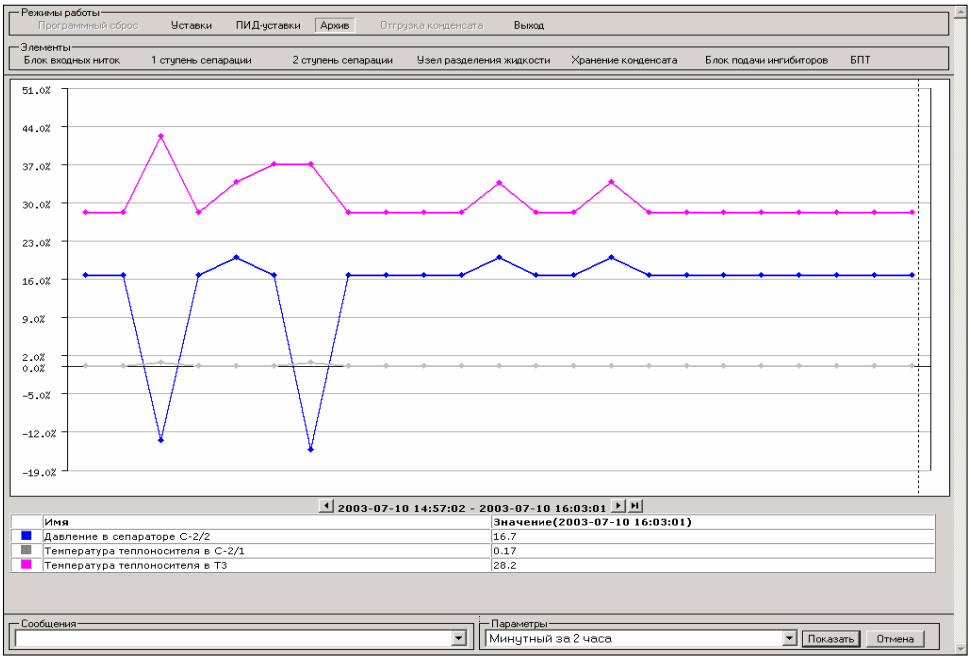

Рис. 5. Графічне представлення архівних даних у ПВАД

ДЕЯКІ ВИСНОВКИ ТА ПЕРСПЕКТИВНІ НАПРЯМКИ ПОДАЛЬШИХ РОЗРОБОК

Розглянуті альтернативні підходи до імплементації візуальних інтерфейсів можуть бути використані в інтранет-системах управління технологічними процесами. Запропоновано уніфіковану архітектуру відповідних програмних сервісів, яка базується на використанні динамічного HTML сумісно з серверним кодом формування даних та дозволяє мінімізувати обмін даними з сервером. Серед напрямків запланованих нами подальших розробок слід виділити використання нотації RDF для моделювання мета-даних у складній розподіленій ІУС; визначення для основних візуальних елементів набору XML-шаблонів, що можуть бути використані повторно із подальшим формуванням на їх основі XML-базованої мови опису візуальних технологічних схем.

ЛІТЕРАТУРА

1. *Рекомендації* науково-практичної конференції «Проблеми автоматизації технологічних процесів та шляхи їх вирішення на підприємствах України», м. Київ, 13–14 листопада 2001 р. — Київ, 2001. — 12 с.
2. *Калядин А.Ю.* SCADA-системы для энергетиков. — М.: ЗАО «РТСофт», 2001. — 156 с.
3. *Apostolov A.* Distribution Substation Protection, Monitoring and Control Systems with Web Browser Based Remote Interface // ALSTOM T&D Protection & Control, Los Angeles, USA. — 2001. — 15 p.
4. *Буч Г., Рамбо Д., Джекобсон А.* UML — руководство пользователя. — М.: ДМК Пресс, 2001. — 432 с.
5. *SCADA-система* «Контур-2» // Киев: ЗАО «Объединение ЮГ», 2003. — 14 с.
6. *Ткачук Н.В.* Перспективная архитектура и информационные технологи для разработки Internet-базированных ИУС АСУ ТП // УСиМ. — 2003. — № 3. — С. 77–83.
7. *Разработка* архитектуры региональной Web-базированной АСУ ТП для объектов газопромыслового управления «Харьковгаздобыча» / Н.В. Ткачук, С.В. Овасапов, Ю.Н. Храпач, К.Н. Щекотихин // Вісник Нац. техн. ун-ту «ХПІ». Тематичний зб. наук. праць «Системний аналіз, управління та інформаційні технології». — Харків: НТУ «ХПІ». — 2002. — **6**, № 9. — С. 51–60.
8. *Ткачук Н.В., Кукленко Д.В.* Применение конценции SCADA-систем для интеллектуального реинжиниринга данных в АСУ ТП // АСУ и приборы автоматики. Всеукр. межвед. науч.-техн. сб. Вып. 121. — 2002. — С. 129–136.
9. *Tkachuk M. V., Mayr H.C. et al*. Web-Based Information Systems For Technological Process Control: Architectural Framework and Software Solutions // Проблемы программирования. — 2002. — № 1–2. — P. 317–325.
10. *Когаловский М.Р.* Энциклопедия технологий баз данных. — М.: Финансы и статистика. — 2002. — 800 с.

Надійшла 10. 04. 2003