UDC 004.9:510.635

Y. M. GONTAR, K. V. TKACH, B. O. YENA, A. V. VASYLENKO

TOWARDS INFORMATION SYSTEM DEVELOPMENT FOR DATA EXTRACTION FROM WEB

Today, the Internet contains a huge number of sources of information, which is constantly used in our daily lives. It often happens that similar in meaning information is presented in different forms on different resources (for example, electronic libraries, online stores, news sites and etc.). In this paper, we analyze the extraction of information from certain type of web sources that is required by the user. The analysis of the data extraction problem was carried out. When considering the main approaches to data extraction, the strengths and weaknesses of each were identified. The main aspects of the extraction of web knowledge were formulated. Approaches and information technologies for solving problems of syntactic analysis based on existing information systems are analyzed. Based on the analysis, the task of developing models and software components for extracting data from certain types of web resources were solving. A conceptual model of extracting data was developed taking into account web space as an external data source. A requirements specification for the software component was created, which will allow to continue working on the project and to clearly understand the requirements and constraints for implementation. During the process of modeling software, the following diagrams have been developed, such as activities, sequences and deployments, which will then be used to create the finished software application. For further development of the software, a programming platform and types of testing (load and modular) were defined. The obtained results allow to state that the proposed design solution, which will be implemented as a prototype of the software system, can perform the task of extracting data from different sources on the basis of a single semantic template.

Key words: information, web search, data extraction, data source, data mining, language standards, informational technology.

Ю. М. ГОНТАР, К. В. ТКАЧ, Б. А. ЄНА, А. В. ВАСИЛЕНКО

ПІДХІД ДО РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЕКСТРАКЦІЇ ДАНИХ З ВЕБ

Сьогодні Інтернет містить величезну кількість джерел інформації, яка постійно використовується в нашому щоденному житті. Часто буває, що схожа за змістом інформація представлена в різній формі на різних ресурсах (наприклад, електронні бібліотеки, інтернет-магазини, новинні сайти). У даній роботі аналізується вилучення інформації з веб-джерел певного типу, яке потрібно користувачеві. Проведено аналіз проблеми вилучення даних. При розгляді основних підходів до екстракції даних були виділені сильні і слабкі сторони кожного. Сформульовано основні аспекти вилучення веб-знань. Проаналізовано підходи та інформаційні технології вирішення проблем синтаксичного аналізу на основі існуючих інформаційних систем. На основі проведеного аналізу була сформована задача розробки моделей і програмних компонентів для отримання даних з веб-ресурсів певного типу. Розроблено концептуальну модель вилучення даних з урахуванням веб-простору як зовнішнього джерела даних. Була створена специфікація вимог для програмного компонента, що дозволить продовжити роботу над проектом, щоб чітко розуміти вимоги і обмеження для реалізації. При моделюванні програмного забезпечення були розроблені наступні діаграми, такі як діаграми класів, активності, послідовності і розгортання, які потім будуть використовуватися для створення готового додатка. Для подальшої розробки програмного забезпечення була визначена платформа програмування і види тестування (навантажувальний і модульне). Отримані результати дозволяють стверджувати, що пропоноване проектне рішення, яке буде реалізовано у вигляді прототипу програмної системи, може виконувати завдання екстракції даних з різних джерел на основі одного семантичного шаблону.

Ключові слова: інформація, екстракція даних, джерело даних,інтелектуальний аналіз даних, язикові стандарти, інформаційні технології.

Ю. Н. ГОНТАРЬ, Е. В. ТКАЧ, Б. О. ЕНА, А. В. ВАСИЛЕНКО

ПОДХОД К РАЗРАБОТКЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ИЗВЛЕЧЕНИЯ ДАННЫХ ИЗ ВЕБ

Сегодня Интернет содержит огромное число источников информации, которая постоянно используется в нашей ежедневной жизни. Часто бывает, что похожая по смыслу информация представлена в разной форме на разных ресурсах (например, электронные библиотеки, интернет-магазины, новостные сайты). В данной работе анализируется извлечение информации из веб-источников определенного типа, которое требуется пользователю. Проведен анализ проблемы извлечения данных. При рассмотрении основных подходов к экстракции данных были выделены сильные и слабые стороны каждого. Сформулированы основные аспекты извлечения веб-знаний. Проанализированы подходы и информационные технологии решения проблем синтаксического анализа на основе существующих информационных систем. На основе проведенного анализа была сформирована задача разработки моделей и программных компонентов для извлечения данных из веб-ресурсов определенного типа. Разработана концептуальная модель извлечения данных с учетом веб-пространства как внешнего источника данных. Была создана спецификация требований для программного компонента, что позволит продолжить работу над проектом, чтобы четко понимать требования и ограничения для реализации. При моделировании программного обеспечения были разработаны следующие диаграммы, такие как диаграммы классов, активности, последовательности и развертывания, которые затем будут использоваться для создания готового приложения. Для дальнейшей разработки программного обеспечения была определена платформа программирования и виды тестирования (нагрузочное и модульное). Полученные результаты позволяют утверждать, что предлагаемое проектное решение, которое будет реализовано в виде прототипа программной системы, может выполнять задачи экстракции данных из разных источников на основе одного семантического шаблона.

Ключевые слова: информация, экстракция данных, источник данных, интеллектуальный анализ данных, стандарты, информационные технологии.

Introduction. The amount of information available through the Internet is constantly increasing. Unfortunately, extracting useful content from this huge amount of data remains an open question. The lack of standard data models and structures makes developers create solutions from scratch. In some cases, the use of descriptions of metadata or data models can help to understand the structure of data.

An expert figure is still needed in many situations where developers do not have the right fundamental knowledge. This forces developers to spend expensive time, absorbing expert knowledge. In other areas there are promising solutions that use machine learning techniques. However, increasing accuracy requires an enlargement the

complexity of the system that cannot be realized in many projects.

The purpose of the study is to develop software components for extraction of data from web resources of a certain type. This requires an analysis of the extraction algorithms, modern programs that are implemented and an overview of the methodology and technology solutions to the problem.

Data extraction. The Internet needs a language that is compatible with the syntax, which allows the user to display data with saving a particular layout. HTML became the language of the Internet and, therefore, the most accepted decision. However, HTML, by default, does not provide any mechanism that facilitates the automatic analysis of existing documents [1]. This restriction does not allow to distinguish content from the layout and semantics of data.

Several standards, such as RDF, RDFS and OWL, have been developed to provide a common syntax for defining data models. These solutions allow to define ontologies that support queries. These technologies are usually not understood by developers who initially ignore the process of semantic annotation in developing HTML pages. To simplify this problem, a later approach called Schema defines a vocabulary of concepts such as people, places, events and products that allow annotation of data contained in an HTML document. This allows to establish a connection between the content and any existing scheme, bring semantics to the Internet.

Web page design can hide data from existing search engines [2]. Using dynamic content, CAPTCHAS, private web pages, scripts or unconnected content among others leads to the creation of Deep Web. A simple example is web pages that use search queries in database. Information contained in the database can't be indexed by the search engine, as this requires the software engine interact with the search form, defines the search parameters, and understands the semantics of the returned data. Commercial search engines, such as Google, Bing, or DuckDuckGo, are developing their tools with a clear focus on indexing the so-called surface network. This makes us think that most of the information contained on the Internet is not indexed. This problem has recently been covered by the MEMEX DARPA project, which shows an attempt to index the information contained in the Deep Web [3].

The processing of data on any of these webpages involves a certain degree of human interaction (filling out the search form, script interaction, etc.). After downloading raw data, it becomes a kind of index format that can be stored in the database for further analysis [4]. When solving the problem of extracting knowledge, most solutions are developed from scratch, engaging in data extraction, analysis and storage. In the case of multiple data sources, the complexity of the problem increases until it becomes impossible.

We can identify three key players in any problem of knowledge extraction. The first is a data source containing relevant information (for example, a web page). The second is a database for data storage purposes (for example, MySQL). The third is an expert who can determine how to translate data into a source and into a database. Transformation between source and base can be considered automatically done. Regardless of the level of automation, the role of an expert is needed to insert some initial semantics of the data before the removal. In addition, the expert is responsible for determining whether data extraction is correct or not. The fourth actor can be developers. The developer is defined as a professional who can contribute to the development of a solution in a technical sense. The developer is appointed to write code, subprograms and programs that prepare processed data.

As a rule, some aspects are ignored in existing projects, such as:

1 Several data sources. Most approaches to data extraction plan deal with several data sources that are similar in design (for example, on Amazon product pages, Wikipedia, etc.). The system must take into account that the data sources do not have a unique design, basically inconsistent in the source or several sources of information.

2 Multilingual approach. Most existing solutions consider only data sources written in one language. For example, Member Countries and partners use their official language for the publication of any EU-related document, and official central reports are available in English, French or German, as the most commonly used languages, by the number of speakers.

3 Extract data from different file formats: HTML is the most common data format. However, other formats such as XML, DOC or PDF may be present. For HTML and XML there are various parsers that also store the formatting structure, so you can use it later. DOC and parser PDFs are harder to find, and most of them extract text data without any formatting. This makes it difficult to automatically extract data. In some extreme cases, files are archived in various formats (for example, ZIP, RAR, self-extracting RAR), and the file preprocessor component must be designed to receive data. Moreover, data encoding is also a problem that depends on the base system. In the current web world UTF-8 is a widely used standard, but in some cases, servers send another encoding from the actual file encoding.

4 Multiple recurring operations and continuous updates. In system design, it's important to keep in mind that data from relevant data sources develops over time, which means that you can add more documents, but the design of the extraction process can also be changed through structural changes to the document. Most of the operations can be reused in domains and data sources, so the conveyor needs to be reused and configurable. Moreover, these operations must be performed continuously and repeatedly, without much human intervention.

The simplest solution adopted by many projects is to use XQuery [5] or regular expressions to extract the exact path to the target element. This approach is not very resistant to structural changes of the document template. Another popular approach is the use of advanced style table language transformations (XSLT) [5], which provides a unified syntax for writing conversion rules from compatible XML languages. In the basic form,

HTML is basically XML-compatible, so this approach can be applied to HTML. This approach is more durable than XQuery to structural changes, but it is usually very difficult to debug. Another simple technique, very practical in small projects, is a simplified version or HTML. This is achieved by simplifying the syntax of HTML by deleting all elements except the main HTML syntax and formatting [6].

In recent years, the problem of extraction of knowledge is intensively studied. The first family of solutions has studied the use of domain languages to determine how data should be deleted. Solutions similar to those presented in use the language of extracting declarative information to determine data extraction plans. Similarly, [7] uses rule sets to determine extraction. In these solutions, the quality of extraction depends in particular on the operators' skills to determine the rules for extracting. The second family of solutions investigates the use of machine learning techniques to improve the information extraction. These solutions are based on using output models that attempt to build relationships for this data set.

An overview of the existing methods of analyzing web content with an emphasis on automated and machine learning methods was reviewed earlier [8]. An automated iterative process for creating a formal description of a document series template using a single, homogeneous template can automatically detect duplicate structures within a single template and create an approximation of the page structure, as well as the ability to obtain any relevant data [9]. IEPAD uses this approach and reduces the complexity of the problem by grouping HTML element tags into different categories [10]. The result of IEPAD is the PAT tree, which is closer to the single, uniform, template document. Van [11] broadens this approach by comparing the similarity of a tree obtained by constructing several tree elements of HTML.

One of the popular frameworks that unifies the methods of machine learning and based on dictionaries is GATE [12]. This toolkit provides a complete structure for annotation, creating dictionaries for named objects, and various methods for processing natural language and machine learning, which is very useful for creating controlled approaches in Data Mining.

Recently, the DeepDive framework [13] has attracted much attention from the research community. DeepDive uses a set of defined rules to establish relationships between objects. The final creation of a database is an iterative loop, in which the operator can control the process of machine learning, identifying errors committed by the system. Similarly, Google Knowledge Vault builds relationships using RDF triplets. This approach shows some similarity to DeepDive with a clear emphasis on data scalability.

Approaches to data extraction from web resources. Extraction is a process of obtaining data from resources, which, as a rule, has a more practical component than the theoretical one. The main purpose of extraction is the collection of data (parsing) with subsequent preservation in the right format. In fact, the task is to write HTML parsers, then it will be discussed in more detail. There are several approaches to extracting data.

DOM tree analysis using XPath. Using this approach, data can be obtained directly by the identifier, name, or other attributes of the tree element (such item can be paragraph, table, block, etc.). In addition, if an item is not marked with any identifier, then you can get it by some unique path, going down the DOM tree or navigating through a collection of similar elements.

Advantages of this approach:

- data of any type and any level of complexity can be obtained;
- knowing the location of an element, you can get its value by writing the path to it.

Disadvantages of this approach:

- various HTML / JavaScript engines generate a DOM tree differently, so you need to bind to a specific engine; the path to the element may change, therefore, as a rule, such parsers are designed for a short period of data collection;
- the DOM path can be complex and not always unambiguous.

This approach can be used in conjunction with the Microsoft.mshtml library, which is essentially the main element in Internet Explorer.

The Data Extracting SDK uses Microsoft.mshtml to analyze the DOM tree, but it is a "superstructure" over the library for ease of use.

The next evolutionary stage of the DOM tree analysis is the use of XPath – that is, the ways that are widely used in parsing XML data. The essence of this approach is to use a simple syntax to describe the path to an element without need for a gradual downward movement of the DOM tree. This approach is well known by the jQuery library and the HtmlAgilityPack library.

Parsing lines. Despite the fact that this approach cannot be used to write serious parsers, it's necessary to pay attention to it.

Sometimes the data is displayed using a certain template (for example, a mobile phone characteristics table) when the values of the parameters are standard and only their values change. In this case, the data can be obtained without analyzing the DOM tree, and by parsing the strings, for example, as it is done in the Data Extracting SDK. The use of a set of methods for analyzing strings sometimes (more often, in simple template cases) is more effective than a DOM tree or XPath analysis.

Using Regular Expressions. Regular expressions should be used only for obtaining data that has a strict format – electronic addresses, telephones, etc., in rare cases – addresses, template data.

Visual approach. At this moment, the visual approach is at an early stage of development. The essence of the approach is that the user could "configure" the system without using a software language or API to get the necessary data of any complexity.

Methods of analyzing web pages at the level of information blocks. There are currently a large number of available web scanners in open source projects. One is the Apache Nutch [14] project, which offers a complete

structure for the development of distributed and scalable scanners that can easily be linked to other solutions from the Apache environment. However, for smaller-scale solutions, other frameworks such as Scrapy are more acceptable. Scrapy lets to define a parallel scanner with Python and provides the developer with a structure that manages simultaneous queries and makes it easier to connect to applications based on Django. To handle pages using JavaScript, the most common approach is to use the Selenium driver [15]. This driver allows you to connect a large number of browsers, such as Firefox or Chrome, using a scanner and emulate the behavior of the user who clicks or text over the current page.

Formulation of the problem. Scanning web pages is the first step in data collection. The items to be traversed may vary, depending on the data source. How to crawl web pages depends on the design of the data source. In some cases, pages are easily accessible through a single URL or can be obtained after completing the search form.

Possible scenarios:

- Identity by public identifier. In this case, each item that is to be traversed is uniquely identified with a URL that contains a unique identifier. If the generation of identifiers is known, you can statically generate a list of possible URLs for the query.
- Identification on an unknown identifier. As in the previous scenario. However, how the IDs are generated is unknown. In this case, the identifiers must first be extracted from the website itself, and then used to create a final destination URL that is to be traversed.
- Dynamic URLs. Many platforms distribute content to dynamic URLs. This makes it impossible to statically generate a list of addresses for study. This script refers to the initial navigation that requests the URL web platform and then generates unique URLs that can identify these elements to ensure their uniqueness in the archival system.
- As mentioned earlier, it's difficult to develop a common scan solution that can be useful in all scenarios. This is due to the fact that many platforms contain a large number of JavaScript code in conjunction with AJAX messaging. It is possible to generalize some aspects that need to be taken into account during the phase of circumvention.
- The following collection of links. Many web portals display content received after a database request. This means that not all of the existing content is displayed at once, but only a small part. One thread can scan each page with a list of links, while others can navigate through existing links.
- Not all items are displayed. The results displayed in these systems are simply the result of a database query. In some cases, the result set is divided into pages that need to be switched, in other cases the resulting set has a limit that prevents full display of existing results. In these cases, it may not be possible to scan all existing results because the system does not disclose this information.
- Massive use of JavaScript. In many cases, the use of JavaScript solutions greatly complicates the scanning of these systems. The easiest solution is to emulate user behavior using Javascript engines. However, it imposes penalties on the side of the scan due to the excessive use of resources of some browsers.
- Limit of queries. To avoid attacks such as "denial of service" (DDoS), many platforms track requests. Exceeding a certain number of queries may result in a temporary termination of the service for the user. Each platform is different, and only a trial error method can reveal what measures these platforms use.
- Cookies and sessions. Many systems use cookies to store session identifiers, which allow the server to identify query parameters that are used by the user. However, these cookies have an expiration date, which can sometimes break the extraction on some platform.
- Not serviced. Bypass servers may fail. Multiple crawling of these servers will simply return the HTTP error code. However, many systems return a web page that informs about the inaccessibility of the service.
- Type of content. The content type code returned by the HTTP header is especially important when parsing web pages that are not encrypted using standard ASCII or UTF8.

The task of this work is to analyze and develop models and software components for extraction of data from web resources of a certain type.

Specification of software requirements. The component developed in this work provides a convenient opportunity for automatic data extraction and its further automatic analysis. The primary area of knowledge is the extraction and analysis of marketing information for obtaining a sample of the most relevant and most advantageous offers in the market. But the component developed in this work has a wide range of uses, from travel agencies to state-owned enterprises.

The purpose of this project is to automate the process of extracting accurate and specific data contained in various WEB-pages. An example of the overall interaction of the system is shown in Figure 1. This system is a complex software and hardware. For its full functioning, the software is developed taking into account the ability to process a large number of requests from users of the product, as well as its reliability, openness for further development, security and properly developed mode for its further support.

This component is developed as a component of the server part of larger software systems. The relationship between the client and the component that is developed on the server use HTTP queries. The request will contain information about which pages to extract, and what type

of data they should provide. On the server component is trying to get information from the page, and if the extraction was successful – analyzes the page. The information that meets the requirements of the user is returned to the client as the response to the request. Communication with the server is carried out using the same HTTP requests, but they only contain a template, created after the analysis, for further use on similar sites.

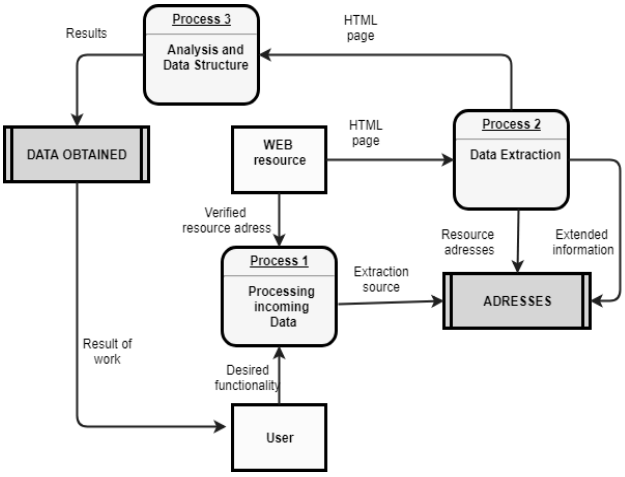

Fig. 1. DFD Diagram of developing system

Figures 2 and 3 depict a component diagram and a use-case diagram respectively.

The system will receive the WEB-address on the input, and return the result to the user in the format that will select in advance. Connection between the interface and the server will use JSON as the serialization format.

Increasing the number of supported resources should not affect performance. Using pre-created and generated templates by the component itself, analyzing pages with increasing number of resources should decrease complexity and consume less time. To implement this requirement, a requirement stability ratio will be used which shows how many of the already implemented requirements have to be processed from release to release in the development of new features. Also, this metric gives an idea of how easy it is to scale the functional of the system, adding new features.

Transferability is implemented through the use of cloud storage. Due to this, it does not matter which server will store the component. If you need to move it from one server to another, the cost should be minimal. The metric for this requirement is the Adaptability Measure, which measures the ability of the system to adapt to requirements changes or re-design of the system or the integration of applications.

Paying attention to the fact that the system can grow to very large sizes, since there are so many online sites, the complexity of the newly configured resources should not be complicated. Due to the configuration of the extractor using YAML files, the developed component can easily adapt to extraction from new a resource without losing a lot of time and effort.

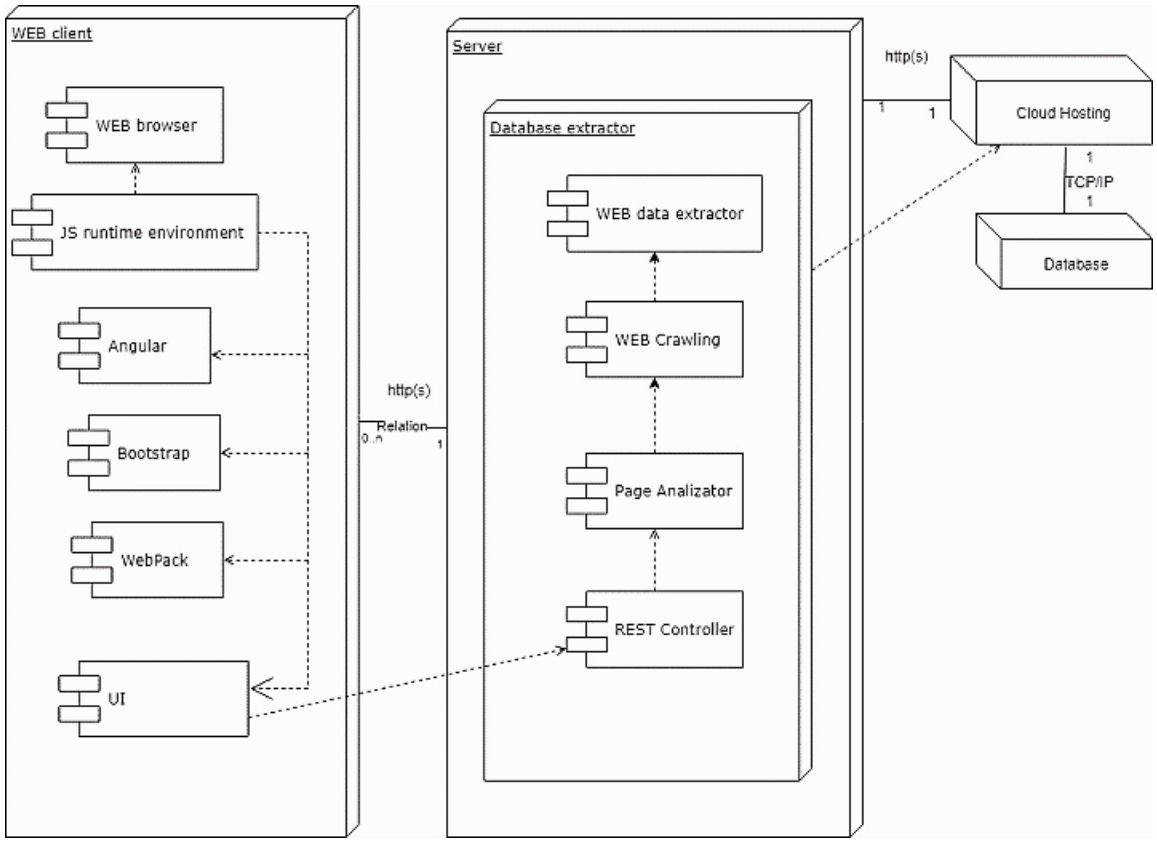

Fig. 2. Component diagram of developing system

Fig. 3. Use-case diagram of developing system

The regression coefficient will be used as a metric for this requirement. The purpose of the metric is to show the efforts of the team: the creation and adjustment of new features or the bulk of time is lost to work with existing parts of the software. The closer the coefficient to 0, the less was the mistakes made to the existing functional in the implementation of new requirements. If the value is greater than 0.5, then we spend more than half of the time on the restoration of earlier functions of the software.

Software development and testing. The main criterion for choosing a platform was the possibility of cross-platform. So, among all the well-known programming languages, the choice falls on Java. In addition to cross-platform, the strengths of the Java language are also high reliability of work, the development of language. The OOP paradigm gives the language such a wonderful feature as scalability, which makes it possible to expand the system many times without difficulty. The Java language is designed to be as reliable as possible. For example, it is prohibited to use multiple imitation in order to avoid ambiguity when referring to the parent class. Instead, the notion of an interface that is no longer a class has been introduced, but contains general guidelines for creating classes and provides multiple imitation.

At the first stage of testing, it is necessary to conduct a modular testing of all components of the system that can be tested separately from the other in the artificial testing environment. Unit testing will be conducted using Junit's Automation Testing Tool. This choice is due to the simple integration of Junit with Java. In the future it is necessary to perform integration testing, which involves testing in two directions. Integration testing of the component level after unit testing is required to verify the correct interaction of parts of the business logic application. This kind of testing can detect errors in the implementation of external interfaces or their improper use. System-level integration testing is needed to find possible mistakes in the interaction between different subsystems of the software, its interaction with the operating system, and other applications.

Load testing is a subset of performance testing, the collection of indicators and the determination of the productivity and time of the response of the software system or device in response to an external request in order to establish compliance with the requirements of this system.

Apache JMeter is a load testing tool developed by the Apache Software Foundation, a Jakarta subproject. Although JMeter was originally developed as a Web application testing tool, it is currently capable of performing load tests for JDBC connections, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP, and TCP.

Postman is a powerful set of API testing tools that has become a must have for many developers. Helps to create test case APIs and improve the productivity of development work. The main purpose of the program is to create collections with API requests.

Locmetrics is very simple freeware. Among the supported languages – C/C++, C#, Java, SQL – it is possible to calculate not only SLOC metrics and its varieties, but also cyclomatic complexity.

Conclusions. An analysis of the data extraction problem was performed. The approaches and information technologies of solving parsing problems on the basis of existing information systems are analyzed. On the basis of the conducted analysis, the task of developing models and software components for extraction of data from web resources of a certain type is set.

The conceptual model of data extraction, considering the web space as an external data source, is developed. This allows to get objective and relevant search results data in the web space.

A requirement specification for a software component has been created. This will allow further work on the project to clearly understand the requirements and limitations to the implementation. The approximate deployment diagram and all types of users that need to be implemented are translated.

In the simulation of the software, the following diagrams were developed such as a class, activity, sequence and deployment diagrams, which will then be used to build a ready-made application.

A software component has been built that allows extraction of data from trading platforms. Its testing and elaborated metrics are responsible for the quality of the product being developed.

Список літератури

1. Baumgartner R., Gatterbauer W., Gottlob G. *Web data extraction system*. In Encyclopedia of Database Systems. 2009. P. 3465–3471.
2. Anupam V., Freire J., Kumar B., Lieuwen D. *Automating web navigation with the WebVCR*. Computer Networks. 2000. P. 503–517.
3. *Memex (Domain-Specific Search).* URL: www.darpa.mil/program/memex (дата звернення: 02.11.2017).

4. Gatterbauer W., Bohunsky P., Herzog M., Krüpl B., Pollak B. Towards domain-independent information extraction from web tables. *Proceedings of the 16th international conference on World Wide Web (May 08–12, 2007, Banff, Alberta, Canada).* New York, ACM, 2007. P. 71–80.
5. Bonifati A., Braga D., Campi A, Ceri S. Active XQuery. *Proceedings of the 18th International Conference on Data Engineering (26 February – 1 March 2002, San Jose, California).* 2002. P. 129–138.
6. Bohannon P., Dalvi N., Filmus Y. Automatic web-scale information extraction. *Proceedings of the ACM SIGMOD ICMD*. 2012. P. 609–612.
7. Shen W., AnHai D., Jeffrey F. Naughton, Ramakrishnan R. Declarative information extraction using datalog with embedded extraction predicates. *In Proceedings of the 33rd International Conference on Very Large Data Bases.* VLDB Endowment. 2007. P. 1033–1044.
8. Crescenzi V. RoadRunner. Towards automatic data extraction from large Web sites. *Proceedings of the 27th International Conference on Very Large Data Bases*. 2001. P. 109–118.
9. Agichtein E., Gravano L. Snowball: extracting relations from large plain-text collections. *Proceedings of the fifth ACM conference on Digital libraries*. 2000. P. 85–94,
10. Arasu A., Garcia-Molina H. Extracting Structured Data from Webpages. *Proceedings of SIGMOD International Conference on Management of Data (June 9–12, 2003, San Diego, California)*. ACM, New York, 2003. P. 337–348.
11. John T. Van Stan, Aron Stubbins, Tree DOM: Dissolved organic matter in throughfall and stemflow. *Limnology and Oceanography Letters.* 2017. Vol. 3. P. 199–214.
12. Cunningham H., Tablan V., Roberts A., Bontcheva K. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS Comput Biol*. 2013. No. 9 (2). P. 31–47.
13. Shin J., Wu S., Wang F., Christopher De Sa, Ce Zhang C., Re C. Incremental knowledge base construction using deepdive. *VLDB Endowment*. 2015. Vol. 8. No. 11. P. 1310– 1321.
14. Khare R, Cutting D., Sitaker K., Rifkin A. Nutch: A Flexible and Scalable Open-Source Web Search Engine. *Proceedings of the 14th International Conference on World Wide Web.* 2005, vol. 1, p. 32.
15. Avasarala S. *Selenium WebDriver Practical Guide.* Pact Publishing, 2014. 266 p.

References

1. Baumgartner R., Gatterbauer W., Gottlob G. *Web data extraction system.* In Encyclopedia of Database Systems. 2009, pp. 3465–3471.
2. Anupam V., Freire J., Kumar B., Lieuwen D. *Automating web navigation with the WebVCR*. Computer Networks. 2000, pp. 503–517.
3. *Memex (Domain-Specific Search).* Available at: www.darpa.mil/program/memex (accessed 02.11.2017).
4. Gatterbauer W., Bohunsky P., Herzog M., Krüpl B., Pollak B. Towards domain-independent information extraction from web tables. *Proceedings of the 16th international conference on World Wide Web (May 08–12, 2007, Banff, Alberta, Canada).* New York, ACM, 2007, pp. 71–80.
5. Bonifati A., Braga D., Campi A, Ceri S. Active XQuery. Proceedings of the 18th International Conference on Data Engineering (26 February – 1 March 2002, San Jose, California). 2002, pp. 129–138.
6. Bohannon P., Dalvi N., Filmus Y. Automatic web-scale information extraction. *Proceedings of the ACM SIGMOD ICMD*. 2012, pp. 609–612.
7. Shen W., AnHai D., Jeffrey F. Naughton, Ramakrishnan R. Declarative information extraction using datalog with embedded extraction predicates. *In Proceedings of the 33rd International Conference on Very Large Data Bases.* VLDB Endowment, 2007, pp. 1033–1044.
8. Crescenzi V. RoadRunner. Towards automatic data extraction from large Web sites. *Proceedings of the 27th International Conference on Very Large Data Bases*. 2001, pp. 109–118.
9. Agichtein E., Gravano L. Snowball: extracting relations from large plain-text collections. *Proceedings of the fifth ACM conference on Digital libraries*. 2000, pp. 85–94,
10. Arasu A., Garcia-Molina H. Extracting Structured Data from Webpages. *Proceedings of SIGMOD International Conference on Management of Data (June 9–12, 2003, San Diego, California)*. ACM, New York, 2003, pp. 337–348.
11. John T. Van Stan, Aron Stubbins, Tree-DOM: Dissolved organic matter in throughfall and stemflow. *Limnology and Oceanography Letters.* 2017, vol. 3, pp. 199–214.
12. Cunningham H., Tablan V., Roberts A., Bontcheva K. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS Comput Biol*. 2013, no. 9 (2), pp. 31–47.
13. Shin J., Wu S., Wang F., Christopher De Sa, Ce Zhang C., Re C. Incremental knowledge base construction using deepdive. *VLDB Endowment*. 2015, vol. 8, no. 11, pp. 1310–1321.
14. Khare R, Cutting D., Sitaker K., Rifkin A. Nutch: A Flexible and Scalable Open-Source Web Search Engine. *Proceedings of the 14th International Conference on World Wide Web.* 2005, vol. 1, p. 32.
15. Avasarala S. *Selenium WebDriver Practical Guide.* Pact Publishing, 2014. 266 p.

Надійшла (received) 25.05.2018

Відомості про авторів / Сведения об авторах / About the Authors

Гонтар Юлія Миколаївна (***Гонтарь Юлия Николаевна***, ***Gontar Yulia Mukolaivna***) – Національний технічний університет «Харківський політехнічний інститут», аспірант; м. Харків, Україна, ORCID: https://orcid.org/0000-0002-3748-5086; e-mail: gontaryn@gmail.com

Ткач Катерина Вікторівна (***Ткач Екатерина Викторовна***, ***Tkach Kateryna Victorivna***) – Національний технічний університет «Харківський політехнічний інститут», студент; м. Харків, Україна, ORCID: https://orcid.org/0000-0001-7104-800X; e-mail: tkachkv@i.ua

Єна Богдан Олександрович (***Ена Богдан Александрович***, ***Yena Bohdan Oleksandrovych***) – Національний технічний університет «Харківський політехнічний інститут», студент; м. Харків, Україна, ORCID: https://orcid.org/0000-0003-4791-956X; e-mail: enafortest@gmail.com

Василенко Артем Вікторович (***Василенко Артем Викторович***, ***Vasylenko Artem Victorovych***) – Національний технічний університет «Харківський політехнічний інститут», аспірант; м. Харків, Україна, ORCID: https://orcid.org/ 0000-0003-3121-4856; e-mail: artyom4ek@yandex.ua