Databases and Information Systems V

Selected Papers from the Eighth International Baltic Conference,
DB&IS 2008

Edited by

Hele-Mai Haav

Institute of Cybernetics at Tallinn University of Technology, Estonia

and

Ahto Kalja

*Department of Computer Engineering of Tallinn University of Technology,
Estonia*

IOS
Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including "Information Modelling and Knowledge Bases" and "Knowledge-Based Intelligent Engineering Systems". It also includes the biennial ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Volume 187

Recently published in this series

Vol. 186. G. Lambert-Torres, J.M. Abe, J.I. da Silva Filho and H.G. Martins (Eds.), Advances in Technological Applications of Logical and Intelligent Systems – Selected Papers from the Sixth Congress on Logic Applied to Technology
Vol. 185. A. Biere, M. Heule, H. van Maaren and T. Walsh (Eds.), Handbook of Satisfiability
Vol. 184. T. Alsinet, J. Puyol-Gruart and C. Torras (Eds.), Artificial Intelligence Research and Development – Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence
Vol. 183. C. Eschenbach and M. Grüninger (Eds.), Formal Ontology in Information Systems – Proceedings of the Fifth International Conference (FOIS 2008)
Vol. 182. H. Fujita and I. Zualkernan (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the seventh SoMeT_08
Vol. 181. A. Zgrzywa, K. Choroś and A. Siemiński (Eds.), New Trends in Multimedia and Network Information Systems
Vol. 180. M. Virvou and T. Nakamura (Eds.), Knowledge-Based Software Engineering – Proceedings of the Eighth Joint Conference on Knowledge-Based Software Engineering
Vol. 179. A. Cesta and N. Fakotakis (Eds.), STAIRS 2008 – Proceedings of the Fourth Starting AI Researchers' Symposium
Vol. 178. M. Ghallab et al. (Eds.), ECAI 2008 – 18th European Conference on Artificial Intelligence
Vol. 177. C. Soares et al. (Eds.), Applications of Data Mining in E-Business and Finance
Vol. 176. P. Zaraté et al. (Eds.), Collaborative Decision Making: Perspectives and Challenges
Vol. 175. A. Briggle, K. Waelbers and P.A.E. Brey (Eds.), Current Issues in Computing and Philosophy

ISSN 0922-6389

Databases and Information Systems V
H.-M. Haav and A. Kalja (Eds.)
IOS Press, 2009

doi:10.3233/978-1-58603-939-4-67

Development of the Plagiarism Detection Tool for Processing Template-Based Documents

Alla ANOHINA [1], Antons MISLEVICS and Janis GRUNDSPENKIS
Department of Systems Theory and Design, Riga Technical University, Latvia

Abstract. The paper presents the plagiarism detection tool for processing template-based documents specifying its conception, the method chosen for the plagiarism detection, the process of the checking of the originality of a submitted student's work, tool's users and requirements. Logical level of the architecture is given as well. Software needed for the implementation and the use of the tool is described.

Keywords. plagiarism detection, Microsoft SharePoint

Introduction

Rapid development of information and communication technologies and growing role of the Internet makes easy available different digital documents inter alia educational materials such as reports and presentations on various topics, full solutions of tasks given in text books, answers of examination questions and even completed course papers and individual works. In this context a serious challenge for the staff of educational institutions is the dissemination of plagiarism. Reasons are threefold [1]. Firstly, this phenomenon is in contradiction to the process of learning which demands from a learner to take certain intellectual and physical efforts in order to acquire knowledge and skills necessary for the further social and professional activities. Secondly, plagiarism reduces the value of a qualification received from an educational institution. Thirdly, it demotivates other students to work independently and to put efforts to learning in case of impunity of plagiarism. In reality students think that those who cheat are rarely caught because their teachers are reluctant to investigate plagiarism. The latter often is true due to the time it takes and burden of proof. And last, but not least, universities apply a wide scale of penalties such as resubmission of the work, reduced mark, or such severe penalty as suspension. All abovementioned sketches the complexity of the plagiarism problem.

The problem of plagiarism is more complicated for teachers teaching courses where individual works are included and the number of students registered for the mentioned courses each year is around several hundreds. So, to avoid plagiarism the teachers must generate several hundreds truly original tasks yearly. Of course, there are some options [2]: random selection of tasks for an individual work from a set of tasks,

[1] Corresponding Author: Alla Anohina, Riga Technical University, Kalku 1, LV-1658 Riga, Latvia; E-mail: alla.anohina@rtu.lv.

variation of input data for each task, permission for students to choose their own problem domains or the use of essay-form assignments. Our experience to use the mentioned approaches simultaneously gives good results: it is possible to generate several hundreds different individual tasks each year, but there are some serious drawbacks, as well. The growing number of individual tasks causes dramatic increase of workload of teachers for checking and assessment of submitted students' works. Moreover, the use of essays puts high cognitive load on learners. These are reasons why we propose to use a template for students' works and a corresponding tool for the detection of plagiarism.

Many different software tools for automated plagiarism detection have been already developed and used, for example, Turnitin, Eve2, CopyCatchGold, WordCheck, Glatt, Moss, JPlag [3, 4, 5, 6, 7, 8]. In general, they provide excellent service for detecting matching text in documents [8]. However our analysis of the mentioned plagiarism detection tools shows that they are inefficient in processing of documents based on a template. In brief, there are two main problems [2]. First, typically plagiarism detection tools check a whole document and as a result legally identical text areas, for example, tasks statements which are the same in all documents are identified as plagiarized parts. It distorts the overall results of the plagiarism detection process and causes difficulties to make decisions about real amount of plagiarism in the document. Second, known plagiarism detection tools compare all parts of a document with all parts of an another document for mutual similarities and as a result semantically unrelated parts of two documents, for example, two solutions of absolutely different tasks are processed unnecessarily. It increases the overall computational time of the document processing.

Previously we have introduced the conception [1] and have gathered and described detailed requirements [2] for a plagiarism detection tool for processing template-based documents. This paper focuses on the architecture of the tool and technologies used for its implementation, and is organized as follows. The first section describes the main idea of the tool, its users, requirements and the checking process of students' works. The second section identifies software needed for the implementation and for the use of the mentioned tool, as well as discusses the logical level of the architecture. Conclusions are given at the end of the paper.

1. Conception and Requirements for the Tool

We define a template is an e-document of pre-designed layout which provides consistent format and content of data put into it and is based on tables and text fields with unique identifiers for input of student's data corresponding to an individual task version. Open XML document format [9] is chosen for templates because it is approved as ISO/IEC International Standard [10] and data in this format can be easily subjected to automated analysis [11].

The main idea of the plagiarism detection tool for processing template-based documents is to make only necessary comparisons according to semantics of document content [1], that is, to compare only semantically correlated parts of documents which potentially can contain identical regions (Figure 1).

1.1. Method of the Plagiarism Detection

One of the popular purely statistical methods, that is, N-gram technique [12, 13] is chosen for the plagiarism detection process in the tool. Before describing the mentioned technique two definitions should be given. A *gram* is a sub-string which has a definite length. A *pattern* is a list of grams extracted from a text.

The comparison of two chunks of text includes two steps: the obtaining of patterns from both chunks of text and the comparison of the acquired patterns.

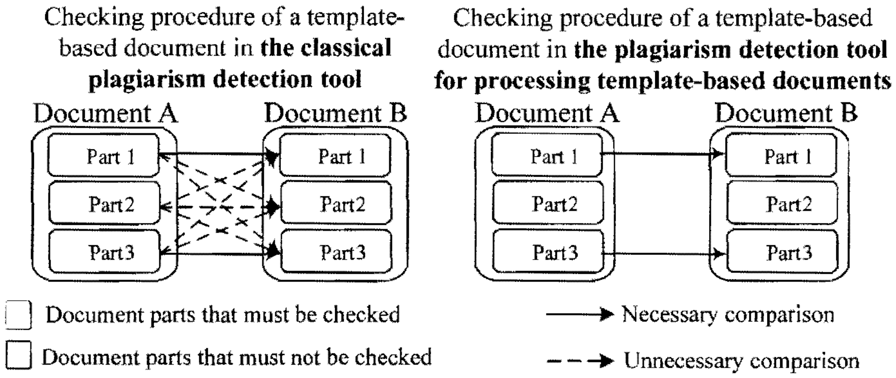

Figure 1. Comparison of documents in classical and template-based approaches to the plagiarism detection (adopted from [1])

The obtaining of a pattern is a process in which sub-strings of a definite length (grams) are extracted from the text. The sub-strings further are used for the comparison of two chunks of text and making of conclusions about their similarity: texts are similar if they have similar grams. Before the extraction of sub-strings the text must be modified in the following way:

- All symbols which do not belong to the alphabet (blank characters, numbers, punctuation signs, and special symbols) must be eliminated from the text.
- All uppercase letters must be replaced by lowercase letters.

For example, in order to extract all 5-grams (sub-strings of 5 symbols) for the sentence "May 15 is my birthday!" the following steps should be performed:

1. Elimination of symbols which do not belong to the alphabet: "Mayismybirthday".
2. Replacing of letters: "mayismybirthday".
3. Extraction of grams: "mayis", "ayism", "yismy", "ismyb", "smybi", "mybir", "ybirt", "birth", "irthd", "rthda", "thday".

When the lists of grams are acquired for both chunks of text they must be compared. This process gives the list with identical grams in both chunks. After that the similarity degree between two texts is calculated using the following equation:

$$SD = ((LP) / (LT)) * 100, \tag{1}$$

where SD is a similarity degree in percentage, LP- the number of identical grams, LT- the number of grams for the document under checking.

1.2. Users of the Tool

In general, the tool must support three groups of users with appropriate access rights and permissions [2]:

- A tool administrator is responsible for the administration of the tool configuration and access rights of other users. The standard functions of the user management, i.e., adding and deleting users, as well as granting them privileges, are widely used in other software systems and therefore are not specified in this paper. Usually administrators of computer classes or departments' laboratory assistants correspond to this group.
- A teacher provides the definition of meta-data and checks the originality of each submitted student's work. As a rule, the role of the teacher plays professors, lecturers or assistants.
- A secretary registers submitted students' works and makes the checking of their formatting against the required format of the template. Typical examples of users of this group are a secretary of the department or of a particular professor, as well as assistants.

However, in the future apart from the three mentioned user groups there are plans to give students an access to the tool in order to reduce the workload of a secretary. Thus, the user groups can be integrated in two main classes: internal users (an administrator, a teacher and a secretary) and external users (students). Figure 2 displays existing and potential users of the tool. All users use a web browser to access the tool.

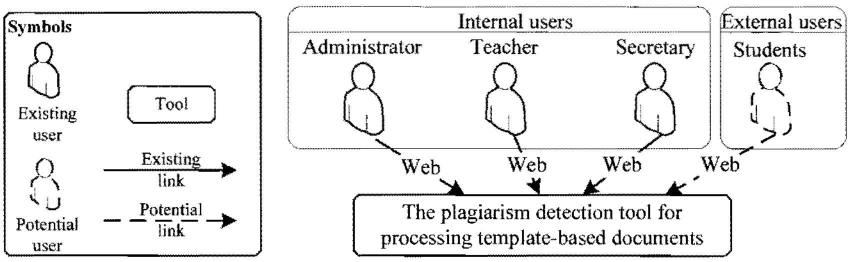

Figure 2. The users of the tool

1.3. Checking of a Student's Work

In general, the checking procedure of a template-based document consists of two stages. Firstly, the document formatting must be checked against the required format of the template. All documents must have the same formatting as defined in the template, otherwise it will not be possible to conduct mutual comparison of documents. Secondly, plagiarized parts of the document under checking must be detected by comparing parts from the document with corresponding parts from other documents.

The detailed steps of the use of the tool are the following [2]:

1. In the preparation step the teacher must describe the structure of the template by defining how many elements (text fields and tables) and with what identifiers the submitted document must contain, as well as, what elements must be compared in documents created on the basis of the same template.

The description of this step can be found in [1]. It will be elaborated after the implementation and evaluation of the first prototype of the tool.

2. The secretary must register the student's submitted work in the tool and must perform the checking of its formatting. As a result of this process the student's work can be recognized as consistent or inconsistent with the required format of the template. In the first case it must be excluded from the further processing. Otherwise, the subsequent steps described below must be applied to the student's work.
3. The teacher must define meta-data of the student's submitted work.
4. The teacher must perform the checking of the originality of the student's submitted work. As a result of this process the student's work can be recognized as containing or not containing plagiarized parts. In the first case the work must be excluded from the further processing. Otherwise, the last step must be applied to the student's work.
5. The teacher must assess the student's work and assign it a mark.

Going through the process described above the student's submitted work can have different statuses [2]:

- "Inconsistent formatting"- the student's submitted work has been recognized as inconsistent with the required format of the template and therefore is excluded from the further processing.
- "Meta-data input"- the student's submitted work has been recognized as consistent with the required format of the template and is waiting when the teacher will define its meta-data.
- "Originality checking"- the student's submitted work has been recognized as consistent with the required format of the template, its meta-data have been defined and it is waiting when the teacher will check it for plagiarism.
- "Plagiarism" - the student's submitted work has been recognized as plagiarism.
- "Non-plagiarism"- the student's submitted work has been recognized as an original work.

Figure 3 displays all processes related to the use of the tool and their actors.

1.4. Requirements for the Tool

Table 1 presents a summary of tool's main functions specifying function purpose, input data, their necessity (mandatory or optional) and source (from the user or from the tool), output data, their necessity (mandatory or optional) and destination (on display or within the tool). Detailed description containing also step-by-step processing of each function is given in [2].

Two main functions of the tool are: the registration of a student's work and checking of its formatting and originality. The first function is performed in the following way [2]:

1. Choosing of the student's work by using a standard file dialog.
2. Input of the date when the student's work was submitted.
3. Checking of the formatting of the submitted work against the required format of the template:
   a) checking of the student's data within the corresponding text fields on the title-page of the student's work: first name, last name, and the number of the student's identity card;

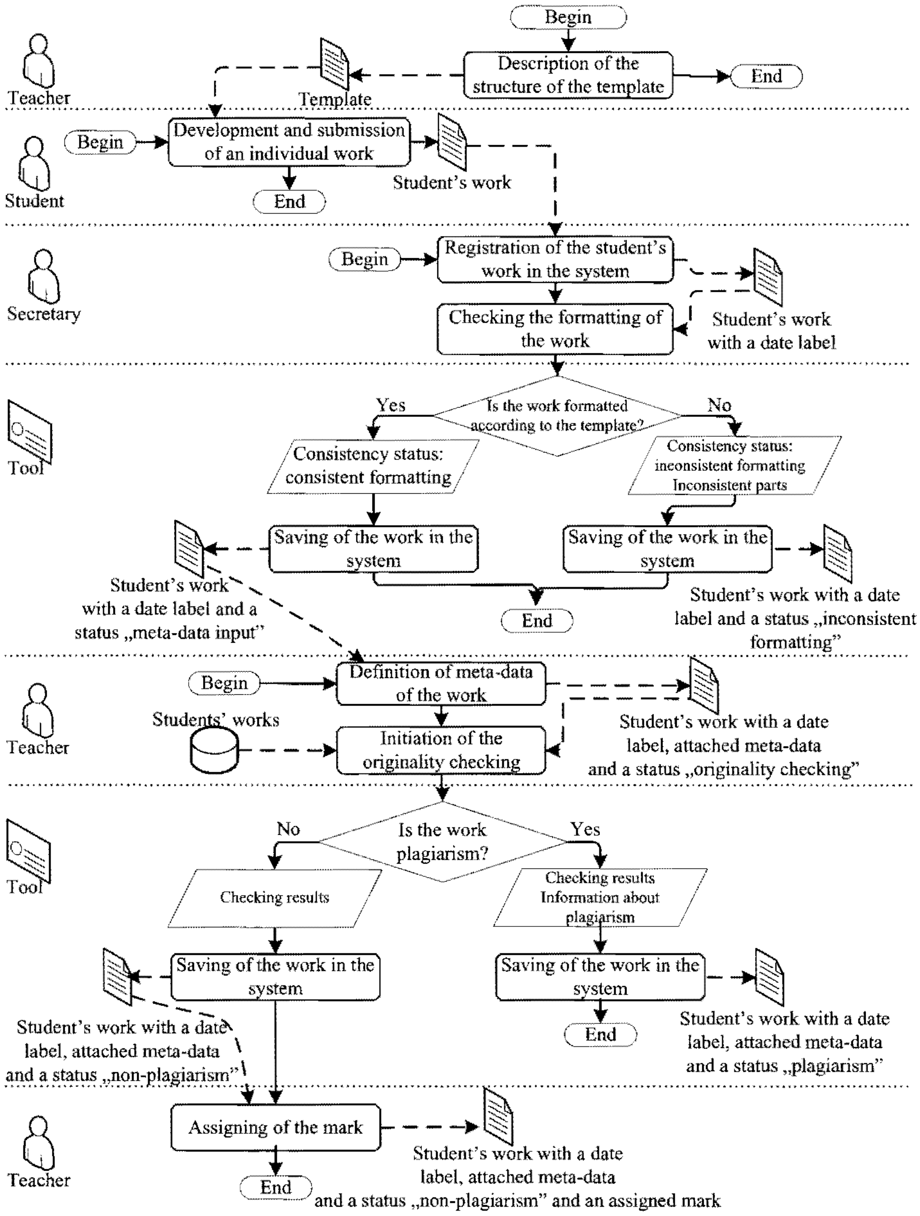

Figure 3. Main processes related to the use of the tool

b) checking whether the student's work contains text fields or tables which are not defined within the template. At this point it is necessary to give additional explanations. As it was mentioned earlier in the paper, the template consists from tables and text fields (elements) with unique

Table 1. Summary of requirements for the tool.

User	Function	Purpose	Input	Output
Teacher	Definition of meta-data	To specify meta-data that give additional information about the student's work and may help in the process of the originality checking	Keywords both for the whole student's work and its particular parts (optional; from the user)	Keywords both for the whole student's work and its particular parts (optional; within the tool)
	Originality checking	To detect either the student's submitted work is an original work or it contains plagiarized parts	Search criteria (optional; from the user) Selected student's works or all students' works registered in the tool (mandatory; from the tool) Student's work under checking (mandatory; from the tool)	Checking results (mandatory; on display): information about the work under checking (student's first name and last name, submission date, and version number) and a list of other students' works which parts are similar to the work under checking Information about plagiarism (optional; within the tool): name of the work which parts are similar with the work under checking and parts of the work where the similarity was detected Status of the student's work under checking: either "plagiarism" or "non-plagiarism" (mandatory; within the tool)
	Archiving/ dearchiving of registered works	To decrease/increase the number of students' works participating in the originality checking	Search criteria (optional; from the user)	Students' works (optional; within the tool)
	Assigning of a mark	To assign a mark to the student's work which is not recognized as plagiarism	Student's work (mandatory; from the tool) Mark (mandatory; from the user)	Student's work (mandatory; within the tool) Mark (mandatory; within the tool)
	Generation of the report on submitted works	To generate the report of students' submitted works selected by using different searching criteria	Search criteria (optional; from the user)	Report on students' submitted works (mandatory; on display)
	Description of the structure of the template	To describe structure of the template	Number and identifiers of elements, as well as comparable pairs of elements (mandatory; from the user)	Description of the structure of the template (mandatory; within the tool)

Table 1 (continued). Summary of requirements for the tool.

User	Function	Purpose	Input	Output
Secretary	Registration of a student's work and checking of its formatting	To register a student's submitted work in the tool and to check its formatting against the formatting of the template. As a result of this process works which have inconsistent formatting are excluded from the further processing	Student's work (mandatory; from the user) Date when the student's work was submitted (mandatory; from the user)	Summary of checking results (mandatory; on display): consistency status (either inconsistent or consistent formatting) and an identification of inconsistent parts of the work Student's work with the status either "inconsistent formatting" or "meta-data input" and date when it was submitted (mandatory; within the tool)
	Generation of the report on submitted works	To generate the report of students' submitted works selected by using different searching criteria	Search criteria (optional; from the user)	Report on students' submitted works (mandatory; on display)

identifiers. The process of the originality checking must compare content of elements with identical identifiers within a pair of documents. If the student replaces some elements in his/her work with new ones or with ones taken from another student's work, identifiers will differ from identifiers of the template. In this case it will not be possible to check the originality of their content in comparison with other documents. Therefore, it is necessary to verify that the student's work does not contain elements with different identifiers before the checking of the originality.

4. Displaying the summary of the checking results by providing information about the student's work consistency with the required format of the template. In case of inconsistency the identification of inconsistent parts of the work must be given.
5. Saving the student's work in the tool's data base:
   a) if the student's work is inconsistent with the required format of the template, it must be stored in the tool's data base by attaching it to the corresponding student with the status "inconsistent formatting";
   b) if the student's work is consistent with the formatting of the template, the tool must generate a unique name for each submission in the form of NoOfStudentIdentityCard_DataOfSubmission_VersionNumber.doc, where NoOfStudentIdentityCard must be taken from the corresponding text fields on the title-page of the student's work, DataOfSubmission is the date provided in the 2nd step of this function, and VersionNumber is a version number of the work for a particular student that the tool must acquire automatically from the data base, taking into account the number of the student's identity card. The student's work must be stored in the

tool's data base by attaching it to the corresponding student with the status "meta-data input".

The originality checking includes the following steps [2]:

1. Selection of students' works which will participate in the checking process. The following criteria can be used: number of the student's identity card, first name and last name, interval of dates when students' works were submitted or a particular date, status of students' works, and a mark. The selection results must be displayed as a list. The user must have the possibilities to select all works from the list or only some of them. The user must have the possibility not to make the selection. In this case the student's work will be compared with all other works registered in the tool.
2. The one-to-one comparison of the documents (the submitted student's work with the selected works) will be made on the basis of the teacher's provided information about what fields must be compared in two documents created on the basis of the same template. The comparison will be performed on the basis of N-gram technique described earlier in the paper (Sub-section 1.1).
3. After the checking process, the tool must display checking results for the user. The results must include information about the work under checking (student first name and last name, number of the identity card, submission date, and version number) and the list of other students' works which parts are similar to the work under checking. The list must contain the information about the name of the work, work parts which are similar with the work under checking, and the similarity degree calculated in accordance with Eq. (1).
4. The possibility to compare students' submissions manually must be provided. In this case the teacher must choose one of the works from the list described in Step 3. The work under checking and the selected work must be displayed in a separate window.
5. The user can mark the students' works in the list described in Step 3 as "source of plagiarism" or "not source of plagiarism". If the work under checking is recognized as plagiarism, the following information must be stored in the tool's data base: name of the work which parts are similar to the work under checking, parts where the similarity was detected, as well as the status of the work under checking must be changed to "plagiarism". Otherwise, the work will acquire the status "non-plagiarism".
6. The possibilities to print out or to save the checking results must be provided.
7. The user will not be allowed to start the checking of another work until all works in the list described in Step 3 will not be marked as "source of plagiarism" or "not source of plagiarism".

2. Design of the Tool

2.1. Software

Microsoft Windows SharePoint Services 3.0 (WSS 3.0) platform is chosen for the implementation of the tool. The platform includes all necessary basic functionality: user authentication, Web access, storing lists of custom information, managing access rights and permissions, search facilities, backup and recovery, document versioning, and workflows. Moreover, the platform provides different integration mechanisms such as WebDAV, Web services, RSS, as well as a rich application programming interface (API) for Microsoft .NET Framework [14, 15]. Additionally, in the future it will be possible to perform the migration of the tool to Microsoft Office SharePoint Server 2007 (MOSS 2007), which extends the WSS 3.0 platform, providing additional features [16, 17].

Figure 4 shows which software must be installed on the server and on clients' computers. The use of WSS 3.0 platform demands the installation of Windows Server 2003 operating system, IIS 6.0 Internet server, Microsoft .NET Framework 3.0, Microsoft SQL Server 2005 and WSS 3.0, or MOSS 2007 on the server side. Most of the popular Web browsers can be used on clients' computers. However, a text editor which supports Open XML document format must be installed in order to provide that students can develop their individual works using the offered template and teachers can create templates.

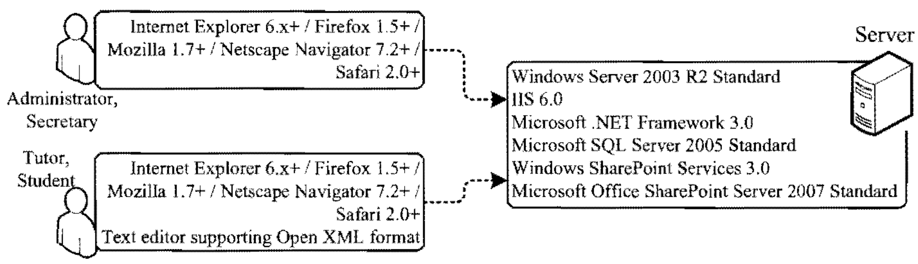

Figure 4. Used software

2.2. Logical Level of the Architecture

Figure 5 displays the logical level of the tool's architecture. All users access the tool through a WSS 3.0 provided authentication layer (2). Internal users (Figure 2) access the tool in the same way as their work computers, by using active directory credentials. Credentials for external users are stored in the database (1) and authentication mechanism validates provided credentials against this database during login procedure. For each class of users (internal or external) a separate WSS zone should be created to ensure the described functionality. All users will work with the tool through the Web interface (3) which also is out of the box WSS 3.0 functionality.

The layer of business logic (5) provides an abstraction from physical elements such as database records and files to logical ones such as objects and documents. This layer will be implemented using WSS objects and API.

Data adapters (6) provide an abstraction of data sources. Here default WSS 3.0 adapters will be used in order to access WSS 3.0 databases, as well as WSS or ADO.NET adapters for accessing the index database. Data will be stored in several storages: students' works (7) and other tool's data in the WSS 3.0 database, indexes of students' works (9) in the WSS 3.0 or an external database (moving indexes to an external database could potentially increase overall performance of the solution).

The work indexer (10) retrieves students' works (7), generates an index needed to search for plagiarism and stores the index in the database (9). This component is a part of the works analyzing agent's functionality (12), thus, links to storages shown in Figure 5 are informative only: data sources will be accessed through an integration layer (4), business logic (5) and data adapters (6).

The searcher of similar works (11) must compare a particular student's work with similar works (7) on the basis of keywords included in meta-data. Indexes of works (9) must be used for the comparison. Similar to the work indexer (10) this component is a part of the works analyzing agent's functionality (12), so the displayed links are informative only.

The integration layer (4) serves as an interface for external systems. In this case WSS 3.0 provided integration mechanisms through API or Web services will be used. API can be used if an external system physically is deployed on the same server, but Web services support integration with systems deployed on other servers as well.

The works analyzing agent (12) searches plagiarism in students' works. The agent will be implemented using C# programming language and will connect to the tool through WSS 3.0 API, because at the first stage of the tool's development it will be deployed on the same server as the tool. The component will be implemented as an external agent that gives the following advantages. First, the agent will access the tool through the authentication layer (2) with its own user name and password. Thus, the tool's administrator will be able to control which data the agent can access. Second, logic of the agent will be fully independent from the tool and it will allow the improvement or the changing of algorithms for the detection of plagiarism without influencing the overall workflow. Third, in case if the number of students' works increases it will be possible to move the agent to another server in order to improve the overall process performance. Fourth, technology for the implementation of the agent is fully independent of technologies used for the development of the tool that provides further improvement of the tool more flexible.

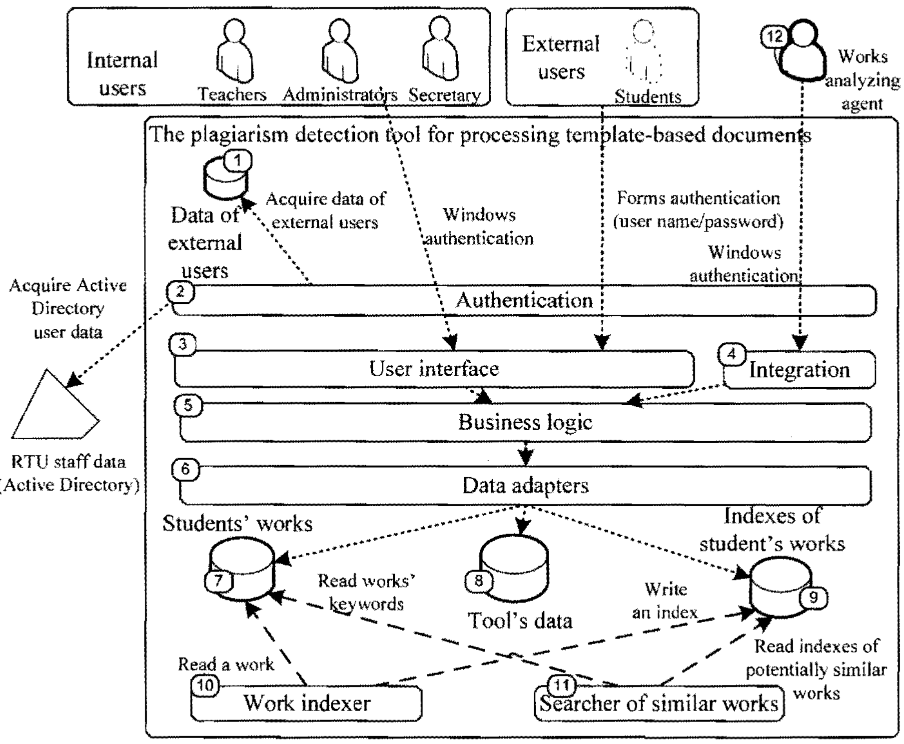

Figure 5. The logical level of the architecture of the tool

3. Conclusions

The problem of plagiarism is investigated in many higher education institutions. Researchers have found wide variations both in rates and in penalties applied. They are trying to find efficient solutions by developing plagiarism detection tools. The analysis of known plagiarism detection tools shows that they are inefficient in processing of documents based on templates. The use of templates is a reasonable approach for decrease of workload of teachers in situations where the number of students who submit their individual works is around several hundreds.

The paper discusses the conception, requirements and design of the plagiarism detection tool which purpose is to process template-based documents. Future work is related to the implementation of the designed tool. The tool will be evaluated experimentally in the study course "Fundamentals of artificial Intelligence".

References

[1] Lukashenko, R., Anohina, A., & Grundspenkis, J. (2007) A Conception of a Plagiarism Detection Tool for Processing Template-Based Documents. In Annual Proceedings of Vidzeme University College, Valmiera, Latvia (in print).

[2] Anohina, A., & Grundspenkis, J. (2008) Requirements of the Plagiarism Detection Tool for Processing Template-Based Documents. In Proceedings of the 8th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2008), Tallinn, Estonia, June 2-5, pp. 51-62.

[3] Maurer, H., Kappe, F., & Zaka, B. (2006) Plagiarism-A Survey, Journal of Universal Computer Sciences, 12 (8): 1050-1084

[4] Delvin, M. (2002) Plagiarism Detection Software: How Effective is it? Assessing Learning in Australian Universities (available at http://www.cshe.unimelb.edu.au/assessinglearning/docs/PlagSoftware.pdf)

[5] Lancaster, T., & Culwin, F. (2000) A Review of Electronic Services for Plagiarism Detection in Student Submissions. In Proceedings of the 8th Annual Conference on the Teaching of Computing, Edinburgh, UK, July 23-25, pp. 54-61.

[6] Lancaster, T., & Culwin, F. (2005) Classifications of Plagiarism Detection Engines, ITALICS, 4 (2) (available at http://www.ics.heacademy.ac.uk/italics/Vol4-2/Plagiarism%20-%20revised%20paper.pdf)

[7] Neill, C.J., Shanmuganthan, G. (2004) A Web-enabled Plagiarism Detection Tool, IT Professional, 6 (5): 19–23

[8] Plagiarism Detection Software Report. (2003) The University of Sydney Teaching and Learning Committee. Draft One (available at http://www.usyd.edu.au/su/ab/docs/2003/ABAgAug03.pdf)

[9] Introducing the Office (2007) Open XML File Formats, (available at http://msdn.microsoft.com/en-us/library/aa338205.aspx)

[10] Open XML Community, (available at http://www.openxmlcommunity.org/)

[11] Open XML Developer, (available at http://openxmldeveloper.org/default.aspx)

[12] Brin, S., Davis, J., & Garcia, M.H. (1995) Copy Detection Mechanisms for Digital Documents. In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25 pp. 398-409.

[13] Tan, C.L, Huang, W., Sung, S.Y., Yu, Z., & Xu, Y. (2003) Text Retrieval from Document Images Based on Word Shape Analysis, Applied Intelligence, 18: 257–270

[14] Windows SharePoint Services 3.0 (available at http://msdn.microsoft.com/en-us/library/bb931737.aspx)

[15] Windows SharePoint Services Developer Center (available at http://msdn.microsoft.com/en-us/sharepoint/default.aspx)

[16] Microsoft Office SharePoint Server 2007 (available at http://msdn.microsoft.com/en-us/library/bb931736.aspx)

[17] SharePoint Server 2007 Developer Portal (available at http://msdn.microsoft.com/en-us/office/aa905503.aspx)