A Massive Data Processing Approach for Effective Trustworthiness in Online Learning Groups

Jorge Miguel[1], Santi Caballé[1], Fatos Xhafa[2], Josep Prieto[1]
[1]Department of Computer Science, Multimedia, and Telecommunication
Open University of Catalonia, Barcelona, Spain
{jmmoneo, scaballe, jprieto}@uoc.edu
[2]Department of Languages and Informatic Systems
Technical University of Catalonia, Barcelona, Spain
fatos@lsi.upc.edu

SUMMARY

This paper proposes a trustworthiness-based approach for the design of secure learning activities in on-line learning groups. Although computer-supported collaborative learning has been widely adopted in many educational institutions over the last decade, there exist still drawbacks that limit its potential. Among these limitations, we investigate on information security vulnerabilities in learning activities, which may be developed in on-line collaborative learning contexts. Although security advanced methodologies and technologies are deployed in learning management systems, many security vulnerabilities are still not satisfactorily solved. To overcome these deficiencies, we first propose the guidelines of a holistic security model in on-line collaborative learning through an effective trustworthiness approach. However, as learners' trustworthiness analysis involves large amount of data generated along learning activities, processing this information is computationally costly, especially if required in real-time. As the main contribution of this paper, we eventually propose a parallel processing approach, which can considerably decrease the time of data processing, thus allowing for building relevant trustworthiness models to support learning activities even in real-time.

KEY WORDS: trustworthiness; e-learning activities; computer-supported collaborative learning; information security; parallel processing; log files; massive data processing, Hadoop, MapReduce

1 INTRODUCTION

Computer-Supported Collaborative Learning (CSCL) has become one of the most influencing educational paradigms [1], [2] widely adopted in many educational institutions over the last two decades. Among these institutions, our real e-Learning context of the Open University of Catalonia[1] (UOC) develops online education based on collaborative learning activities. This institution is supporting the research work presented in this paper and its results are considered and included in other UOC's research projects, with the aim of enhancing e-Learning factors, such as assessment cost reduction and students scalability. Although CSCL activities have been incorporated in many on-line educational settings, there exist still many drawbacks that limit their potential. Among these limitations, collaborative learning services and activities are usually designed and implemented without much consideration of security issues. As a result, information security vulnerabilities may interfere in these activities, thus threatening and reducing the effectiveness of the overall collaborative learning process [3], [4].

Information security requirements have been generally considered and developed recently in Learning Management Systems (LMS) [5]. However, to the best of our knowledge, integrated and holistic security models have not been carried out yet. As a result, many security vulnerabilities are still reported in LMSs and remain unsolved [6], [7]. Therefore, innovative security solutions are needed to overcome these limitations and support a secure learning process. To this end, in this paper we propose a trustworthiness

model based on a multi-fold assessment approach of CSCL activities, which can meet security requirements of on-line collaborative learning process.

Finally, in order to provide effective and just-in-time trustworthiness information from the LMS, it is required a continuous processing and analysis of group members' interaction data during long-term learning activities, which produces huge amounts of valuable data stored typically in server log files [8], [9]. CSCL activities may demand a great amount of communication processes, collaborative contents and many types of interactions [1], [2]; if our model aims to analyze how trustworthiness factors are related to these resources, the context of CSCL will be an ideal case study. Due to the large or very large size of data generated daily in online learning activities, the massive data processing is a foremost step in extracting useful information and may require computational capacity beyond that of a single computer [10]. We study the feasibility of a parallel approach for processing large log data files of a real LMS using distributed infrastructures and show how considerable improvements in performance can be achieved via Hadoop MapReduce implementations.

The paper is organized as follows. Section 2 presents the background and context information on security in e-Learning. Section 3 endows our security model with trustworthiness properties on learning activities describing relevant trustworthiness factors and rules that have an effect in the collaborative learning process. Parallel processing paradigms are analyzed in section 4 to massive data processing and build relevant trustworthiness models. Finally, Section 5 concludes the paper highlighting the main findings and outlining ongoing and future work.

2 Background

In this section we first review main works in the literature on general security in e-Learning, including our previous research. Then, we propose complementary solutions to secure e-Learning beyond technological approaches. To this end, a trustworthiness approach for secure e-Learning is provided.

2.1 Information Security in e-Learning

Early research works about information security in e-Learning [11], [12] are focused on confidentiality issues with respect to ensure students' and tutors' privacy requirements. An initial work [13] suggests that the most effective mechanism for dealing with the privacy issues raised in the virtual learning environment should be a task force or committee made up of those who are closely involved. This proposal is quite general and then in subsequent works on privacy in e-Learning some authors have addressed the need for more specific approaches [3]. Further works [4], [14] consider other aspects of security in e-Learning. In [14], the author argues that security is mainly an organizational and management issue and improving security is an ongoing process in e-Learning. This is in fact the first proposal in which information security is applied to learning management systems as a general requirement in e-Learning design and management. The authors in [4] presented how security in e-Learning can be analyzed from a different point of view, namely by first analyzing threats for e-Learning and then, recommendations are introduced and discussed in order to cope with detected threats. Finally, more specific security issues in e-Learning have been investigated (e.g. virtual assignments and exams, security monitoring, authentication and authorization services) in [15]–[18].

Although the above literature discuss on security design in e-Learning from a theoretical point of view, there is still needful to understand attacks in order to discover security design factors and figure out how security services must be classified and designed [19]. In [20], through analyzing existing research in attack classification, a new attack taxonomy is constructed by classifying attacks into dimensions. Nevertheless, since attacks taxonomies might be applied to cover each kind of attack which might occur in LMS they are not closely related to security design in e-Learning. In order to fill this gap, in [17], we have proposed an alternative approach which associate attacks to security design factors.

There is an increasing interest in understanding security attacks in real life scenarios. Several reports justify the relevance of security attacks during the last two years. In particular, the study presented in [7] revealed that security attacks are a reality for most organizations: 81% of respondents' organizations experienced a security event (i.e. an adverse event that anyhow compromises security). Finally, we can consider specific LMS real software vulnerabilities. Moodle is an Open Source LMS which is massively deployed in many schools and universities. In Moodle Security Announcements[2], 40 serious vulnerabilities have been reported in 2013.

2.2 Previous Work on Security in e-Learning

In previous research [15]–[18] we have argued that general security approaches proposed so far do not guarantee that learning processes are developed in a reliable way. Next we summarize the main research findings on security in e-Learning made so far focused mainly in the following educational contexts: collaborative learning (CSCL), mobile learning (m-Learning), and massive open online courses (MOOCs). These contexts are approached by several design methodologies and security considerations, such as, software modeling languages, risk management, security in LMS, attacks in e-Learning, students' privacy, specific security properties, e.g. authentication, and global user authentication services.

In [15] we proposed a new approach named Secure Collaborative Learning Management Systems (SCLMS) based on an the current developments in the domain of CSCL systems that consider security as a key requirement. As a result, an innovative guideline is proposed to develop secure learning management systems focusing on the support for CSCL with specific needs, such as interactions between participants, collaborative material management, communication processes and generation of collaborative results. Following the SCLMS roadmap, in [17] an innovative guideline to develop secure e-Learning systems was presented for m-Learning. In [18] we conducted research to provide information security to the Massive Open Online Courses (MOOCs) [18] and in particular supporting evaluation, grading and certification as the main challenges in the MOOC arena [21]. The core of this approach is an authentication service defined as a modular PKI-based security model called MOOC Smart Identity Agent (MOOC-SIA) [18], which is a global user authentication model for MOOC platforms.

Considering the previous research experiences, the starting point of this paper is to extend our above proposals with a new trustworthiness model.

2.3 Trustworthiness and Security for e-Learning

In [22] it is discussed that security is both a feeling and a reality. The author points out that the reality of security is mathematical based on the probability of different risks and the effectiveness of different countermeasures. But, security is also a feeling, based not on probabilities and mathematical calculations, but on your psychological reactions to both risks and countermeasures [22]. This security model eventually concludes that security is a trade-off between the real fact that absolute security does not exist and the need to feel secure. This approach is very relevant in our model because it is based on a hybrid evaluation system in which technological and trustworthiness solutions are combined.

In order to measure trustworthiness and identify what factors are involved in a quantitative study, in [23] it is proposed a data provenance trust model, which assigns trust scores to both data and data providers based on certain factors that may affect trustworthiness. For instance, in our e-Learning context, students and students' resources (e.g. shared documents, posts in a forum, etc.) can be modeled following this approach when developing CSCL activities. To this end, in [24], the author designs a survey to explore interpersonal trust in work groups identifying trust-building behaviors ranked in order of importance. These behaviors can be used as trustworthiness factors, which can measure trust in those activities that students develop. In addition, [25] considers different aspects of trustworthiness in terms of expressions and classifications of trust characteristics, such as trust asymmetry, time factor, limited transitivity and reliability.

3 METHODOLOGY

This section shows a methodological approach to build our trustworthiness-based security model for CSCL. First, we built our model by enhancing standard security models with trustworthiness factors and rules, following the considerations made in Section 2. Then, we apply some statistical techniques to the variables involved in our security model with the purpose to measure correlation. Finally, we conclude the section with important issues concerning the management of the large and complex data forming our model, which becomes the main motivation of this research. Next section presents a solution to these issues.

3.1 Trustworthiness for Secure e-Learning

Our security model is endowed in this subsection with trustworthiness properties on learning activities and learners themselves. First, we describe relevant trustworthiness factors and rules that have an effect in the collaborative learning process. Then, in order to measure the impact of these factors, we propose several indicators and levels of trustworthiness.

3.1.1 Trustworthiness Factors and Rules

The relevant trustworthiness factors identified are summarized in the following table:

Table I: Trustworthiness Factors

	Trustworthiness Building Factors (TBF) Student "S" working in the group of students "GS" is building trustworthiness when…
1	S communicates honestly, without distorting any information.
2	S shows confidence in GS's abilities.
3	S keeps promises and commitments.
4	S listens to and values what GS say, even though S might not agree.
5	S cooperates with GS and looks for mutual help.
	Trustworthiness Reducing Factors (TRF) Student "S" working in the group of students "GS" is reducing trustworthiness when…
1	S acts more concerned about own welfare than anything else.
2	S sends mixed messages so that GS never know where S stands.
3	S avoids taking responsibility.
4	S jumps to conclusions without checking the facts first.
5	S makes excuses or blames others when things do not work out.

In addition, we take into account the following trustworthy rules: (i) Asymmetry, where A trust B is not equal to B trust A; (ii) Time factor, where trustworthiness is dynamic and may evolve over the time; (iii) Limited transitivity, where if A trusts C who trusts B then A will also trust B, but with the transition goes on, trust will not absolutely reliable; (iv) Context sensitive, when if context changes, then trust relationship might change too.

However, it is worth mentioning that trustworthiness factors are defined from the perspective of students' behavior, hence the methods discussed so far provide security improvements but cannot fully meet secure e-Learning activities requirements. Furthermore, neither trustworthiness nor PKI models define or manage the actions to take when the security service detects either anomalous situations or violation of the properties we have defined.

Trustworthiness building and reducing factors are closely related to (see Table I):

- Interactions between participants (e.g. TRF2).
- Content management and generation of collaborative results (e.g. TRF1).
- Communication processes (e.g. TBF1).
- Group management tasks (e.g. TRF5).

Every of these issues may be involved in e-Learning, but in CSCL learning experiences, we can find a higher amount of them than in other learning paradigms hence we focus our trustworthiness model on CSCL.

3.1.2 Modeling Trustworthiness Levels and Indicators

We introduce now the concept of trustworthiness indicator tw_i (with $i \in I$, where I is the set of trustworthiness indicators) as a measure of trustworthiness factors. Trustworthiness factors have been presented as those behaviors that reduce or build trustworthiness in a collaborative group and they have been considered in the design of questionnaires. A tw_i is associated with one of the measures defined in each e-assessment instrument (i.e. ratings, questionnaires, reports, etc.). The concept of trustworthiness level Ltw_i is a composition of indicators over trustworthiness rules and characteristics. For instance, we can consider two trustworthiness indicators (tw_a and tw_b). These indicators are different, the first indicator could be a rating in a forum post and the second one a question in a questionnaire; but they measure the same trustworthiness building factor (e.g. TBF-1: communicates honestly). With regarding to trustworthiness rules, this indicator may be compared to the group, over the time or considering the context. Trustworthiness indicators can be represented following these expressions:

$$tw_{a_{r,s}}\ a \in \{Q, RP, LGI\}, r \in R, s \in S$$

where Q is the set of responses in Questionnaires, RP is the analogous set in Reports, LI is the set of LMS indicators for each student (i.e. ratings and the general students' data in the LMS). S is the set of students in the group and R is the set of rules and characteristics (e.g. time factor). These indicators are described above when presenting instruments.

Once indicators have been selected, trustworthiness levels can be expressed as follows:

$$Ltw_i = \sum_{i=1}^{n} \frac{tw_i}{n}, i \in I$$

where I is the set of trustworthiness indicators which are combined in the trustworthiness level Ltw_i.

Trustworthiness levels Ltw_imust be normalized; to this end, we have reviewed the normalization approach defined in [26] with regarding to support those cases in which particular components need to be emphasized more than the others. Following this approach, we previously need to define the weights vectors:

$$w = (w_1, \dots, w_i, \dots w_n), \sum_{i}^{n} w_i = 1$$

where n is the total number of trustworthiness indicators and w_i is the weight assigned to tw_i.

Then, we define trustworthiness normalized levels as:

$$Ltw_i^N = \sum_{i=1}^{n} \frac{(tw_i * w_i)}{n}, i \in I$$

Therefore, trustworthiness levels allow us modeling students' trustworthiness as a combination of normalized indicators using research and data gathering instruments.

Regarding groups, this model may also be applied in cases with only one working group; in this scenario, all students would belong to the same group.

3.2 Statistical Analysis

Following the trustworthiness model presented above we proceed now with inquiring whether the variables involved in the model are related or not. With this purpose the correlation coefficient may be useful. Some authors have proposed several methods regarding rates of similarity, correlation or dependence between two variables [27]. Even though the scope of this paper is focused on user-based collaborative filtering and user-to-user similarity, the models and measures of the correlations between two items applied in this context are completely applicable in our scope. More precisely, we propose Pearson correlation coefficient *r* as a suitable measure devoted to conduct our trustworthiness model. Pearson coefficient applied to a target trustworthiness indicator is defined below:

$$r_{a,b} = \frac{\sum_{i=1}^{n}(tw_{a,i} - \overline{tw}_a) * (tw_{b,i} - \overline{tw}_b)}{\sqrt{\sum_{i=1}^{n}(tw_{a,i} - \overline{tw}_a)^2} * \sqrt{\sum_{i=1}^{n}(tw_{b,i} - \overline{tw}_b)^2}}$$

where tw_a is the target trustworthiness indicator, tw_b is the second trustworthiness indicator in which tw_a is compared (i.e. similarity, correlation, anomalous behavior, etc.), $\overline{tw}_a$ and $\overline{tw}_b$ are the average of the trustworthiness indicators and *n* is the number of student's provided data for tw_a and tw_b indicators.

It is worth mentioning that if both *a* and *b* are trustworthiness indicators with several values over the time (e.g. a question which appears in each questionnaire), they must be compared in the same point in time. In other words, it is implicit that $r_{a,b}$ is actually representing r_{a_t,b_t}, where a_t is the trustworthiness indicator in time *t*.

In addition, this test may be applied to every trustworthiness indicator taking one of them as target indicator. To this end, we define the general Pearson coefficient applied to a target trustworthiness indicator over the whole set of indicators, as follows:

$$r_A = (r_{a,1}, \dots, r_{a,i}, \dots, r_{a,n-1}), i \in I, i \neq a$$

where $r_{a,i}$ is the Pearson coefficient applied to a target trustworthiness indicator is defined above and *I* is the set of trustworthiness indicators.

Both relation and similarity are represented by $r_{a,b}$ and r_A grouping students' activities and taking the variables at the same time. We are also interested in time factor and it may be relevant the evolution of trustworthiness indicators throughout the course. To this end, we extend pervious measures, adding time factor variable:

$$r_{a,t,tt} = \frac{\sum_{i=1}^{n}(tw_{a_{t,i}} - \overline{tw}_{a_t}) * (tw_{a_{tt,i}} - \overline{tw}_{a_{tt}})}{\sqrt{\sum_{i=1}^{n}(tw_{a_{t,i}} - \overline{tw_{a_t}})^2} * \sqrt{\sum_{i=1}^{n}(tw_{a_{tt,i}} - \overline{tw_{a_{tt}}})^2}}$$

where t is the target point in time and tt is the reference point in time (i.e. t is compared against tt), all other variables have already been defined with this case they are instanced in two moments in the course.

Similarly, we can calculate $r_{a,t,tt}$ for each tt, and then the following indicator may be used:

$$r_{A,t} = (r_{a,1}, \dots, r_{a,i}, \dots, r_{a,n-1}), i \in I, i \neq a$$

The trustworthiness indicators are summarized in the Table II:

Table II: Trustworthiness Basic Indicators

Basic Indicators	**Trustworthiness Statistical Analysis**		
	Description	***Group by***	***Target/ Reference***
$r_{A,b}$	Pearson coefficient applied to a target trustworthiness indicator.	Students	tw_a tw_b
r_a	$r_{a,b}$ over the set of indicators	Indicators	tw_a
$r_{a,t,tt}$	Pearson coefficient applied to a tw indicator throughout the course from t to tt.	Time	tw_a t
$r_{A,t}$	$r_{a,t,tt}$ over the throughout the course.	Course	tw_a

Finally, trustworthiness indicators may be gathered in a trustworthiness matrix with the aim of representing the whole relationship table for each indicator:

$$R_{tw} = \begin{bmatrix} 0 & r_{tw_1,tw_2} & \dots & r_{tw_1,tw_n} \\ 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & r_{tw_{n-1},tw_n} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Although the indicators presented are proposed as suitable options for our model, the model will be refined to select those indicators oriented to perform the best similarity and correlation. In addition, this approach is also intended to be a prediction tool, since similarity facts may conduct predictions about the evaluation system and its evolution.

To sum up, the above indicators, levels, rules and statistical analysis can become robust instruments to appropriately modeling trustworthiness in e-learning groups and eventually extending current security models for CSCL that overcomes many of the limitations reported in Section 2. However, the collection of valuable data and their later statistical analysis to build our security model usually involves the constant processing and analysis of large amounts of ill-formatted information, even in real time, stressing even more the computational cost involved and requiring a high performance solution to alleviate this cost. For instance, in our real e-learning context of the UOC, with thousands of on-line courses and many of them involving e-learning in work teams, the amount of data collected can be of the scale of 20GB per day coming from different LMS with different formats, and the information is found with high degree of redundancy, tedious and ill-formatted as well as incomplete.

Next section presents our parallel data processing approach to overcome this problem in order to make it feasible the construction of security models, such as our trustworthiness-based security model for CSCL presented above.

4 PARALLEL PROCESSING APPROACH

In this section we address the need to alleviate the computational cost of massive processing of the large amounts of data generated during long-term e-learning activities, with the aim to cope with learner's trustworthiness analysis and the building of trustworthiness models, even in real-time. To this end, we propose a parallel approach for massive data processing.

4.1 The Problem of Processing Log Files

In previous research [28], [29], [10] we showed that extracting and structuring LMS log data is a prerequisite for later key processes such as the analysis of interactions, assessment of group activity, or the provision of awareness and feedback involved in CSCL. With regarding to BSCW, the computational complexity of extracting and structuring BSCW log files is a costly process as the amount of data tends to be very large and needs computational power beyond of a single processor (see Fig. 1-A and also [10], [29]). In addition, in [30] we studied the viability of processing very large log data files of a real virtual campus (UOC Virtual Campus) using different distributed infrastructures to examine the time performance of massive processing of log files. It was also shown the linear execution time of the local processing of UOC log files (see Fig. 1-B); hence the computational cost of sequentially processing large amounts of log data becomes unfeasible.

Therefore, parallel techniques to speed and scale up the structuring and processing of log data are required dealing with log data. In [28] and [30] these models were implemented following the master-slave paradigm and evaluated using Cluster Computing and Planet Lab platforms.

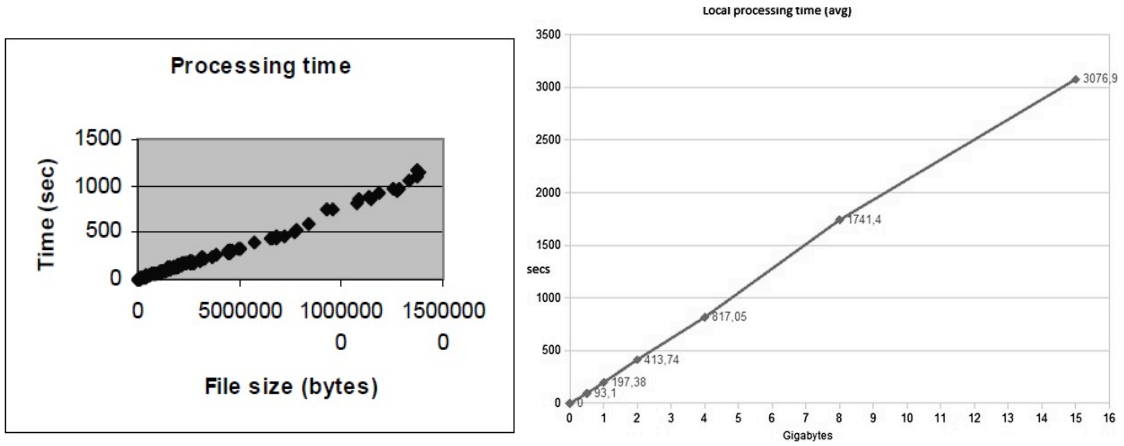

Figure 1. Sequential processing of BSCW log files (A) [10] and local processing of UOC logs (B) [30].

Taking these approaches as starting point, in this paper we extend our goals in two different directions which are presented in next sections: parallelizing the normalization of several LMS logs files (e.g. BSCW and UOC log files) and using MapReduce paradigm [31]. Then, we use Hadoop and Cluster Computing to implement and evaluate the parallelization of massive processing of log data [8].

4.2 UOC Virtual Campus Log Files

Before presenting our parallel processing implementation details, we first show in this section the different format of BSCW and UOC log files and the problems to process them due to the large size and ill-structure formats of both. To this end, a normalization approach for both types of log data is proposed as an input to our general parallel processing model presented in next subsections.

4.2.1 BSCW Log Files

In our real learning context of the Open University of Catalonia, several on-line courses are provided involving hundreds of undergraduate students and a dozen of tutors in a collaborative learning environment. The complexity of the learning practices entails intensive collaboration activity generating a great amount of group activity information. To implement the collaborative learning activities and capture the group interaction we use the abovementioned BSCW as a shared workspace system, which enables collaboration over the Web by supporting document upload, group management and event service among others features. BSCW event service provides awareness information to allow users to coordinate their work [32].

In the BSCW, the events are triggered whenever a user performs an action in a workspace, such as uploading a new document, downloading (i.e. reading) an existing document, renaming a document and so on. The system records the interaction data into large daily log files and presents the recent events to each user. In addition, users can request immediate email messages whenever an event occurs, and the daily activity reports are sent to them daily and inform them about the events within the last 24 hours. The typical format of the BSCW log files is as follows:

```
User:[3434841, '*******']
object:[3452718, 'Presentació A**** S*****']
Type:RateEvent
Time:1078202945.04
Members:[[3448332, '******', 'OyvLkYg2ueStI'], [3449370, '*****', …., [3425007, 'Aula 5 (*****)'], [3425034, 'Espai per a la Formació de Grups'],
[3425118, 'Espai Presentacions']]
On:[3425118, 'Espai Presentacions']
Touched:[3434844, ':********']
Icon:'/bscw_resources/icons/e_write.gif'
Class:Document
Content:application/octet-stream
```

The BSCW log does not follow a standard log format therefore parsing these logs format requires a customized development. Moreover, relevant data are omitted, in the example above the student 3434841 is rating the resource 3452718 but we cannot find additional information such as the rate value.

4.2.2 UOC Log Files

The Web-based Virtual Campus of the UOC is made up of individual and community virtual areas such as mailbox, agenda, classrooms, library, secretary's office and so on. Students and other users (lecturers, tutors, administrative staff, etc.) continuously browse these areas where they request for services to satisfy their particular needs and interests. For instance, students make strong use of email service so as to communicate with other students and lecturers as part of their learning process. All users' requests are chiefly processed by a collection of Apache[3] web servers as well as database servers and other secondary applications, all of which provide service to the whole community and thus satisfy a great deal of users' requests. For load balance purposes, all HTTP traffic is smartly distributed among the different Apache web servers available. Each web server stores in a log file all users' requests received in this specific server and the information generated from processing the requests. Once a day (namely, at 01:00 a.m.), all web servers in a daily rotation merge their logs producing a single very large log file containing the whole user interaction with the campus performed in the last 24 hours. A typical daily log file size may be up to 20 GB. This great amount of information is first pre-processed using filtering techniques in order to remove a lot of futile, non relevant information (e.g. information coming from automatic control processes, the uploading of graphical and format elements, etc.). However, after this pre-processing, about 2.0 GB of potentially useful information corresponding to 5,000,000 of log entries in average still remains [30].

The log files storing the whole activity of the UOC Virtual Campus follow the Apache log system. A typical configuration for the Apache log system is the Common Log Format [33]; a standard configuration for this log system is as follows:

$$LogFormat\ "\%h\ \%l\ \%u\ \%t\ \backslash"\%r\backslash"\ \%>s\ \%b"\ common$$

where (h) is the IP address of the client or remote host; (l) indicates unavailable requested information; (u) is the user id; (t) is the time that the server finished processing the request; (r) is the request line; (s) is the status code and (b) is the size of the object returned.

In UOC Virtual Campus, log file records are managed following a variation of the Common Log Format known as Combined Log Format [33], with two additional fields:

$$LogFormat\ "\%h\ \%l\ \%u\ \%t\ \backslash"\%r\backslash"\ \%>s\ \%b\ \backslash"\%\{Referer\}i\backslash"\ \backslash"\%\{User-agent\}i\backslash""\ combined$$

where (Referer) field shows the site that the client reports having been referred from, and (User-agent) field identifies information that the client browser reports about itself.

As an example, the following is a record that is part of a real log of the UOC Virtual Campus (IP address has been anonymized):

[15/Mar/2012:00:26:40 +0100] xxx.xxx.xxx.xxx "POST /WebMail/listMails.do?mensajeConfirmacion= El%20missatge%20s´ha%20desplaçat%20a%20la%20carpeta%20Rectorat HTTP/1.1" 200 "http://cv.uoc.edu/WebMail/readMail.do" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; OfficeLiveConnector.1.3; OfficeLivePatch.0.0; InfoPath.2; BRI/2)" 8857 20A

This record example illustrates that the user id parameter described in Apache Combined Log Format is not available in this line. Moreover, both unavailable requested information (*l*) and the size of the object (*b*) do not meet the standard arrangement. Although user identifications are not stored in log files, the system maintains a session id, this value is a user session key (a 128-character string long) included as a parameter in the request.

At this point, we highlight certain problems arisen by dealing with these log files:

- We can identify uniquely neither the user nor the record.
- Each explicit user request generates at least an entry in the log file but it usually generates additional requests, for instance, in order to compose a user web interface, each component (i.e. image, style sheets, etc.) will be loaded using GET operations. This information is not relevant and these records unnecessarily increase both the storage space and processing effort.
- Additional parameters introduced in Combined Format (Referer and User-agent) may be useful for audit purposes, but in our context this values introduce a high degree of redundancy.

Because of these problems must be solved, we propose several actions. In the case of the user identification, this limitation cannot be completely solved because this information is unavailable, but regarding records identification it is possible to combine several fields as a record key. The parameters selected to identify a record are:

$$Record_{Key} = \{IP; Time; Session\}$$

Redundant and unnecessary information must be parsed and ignored. To this end, these actions have been implemented in the Java class *Action*, which is described in the following subsection following a record taxonomy devoted to clean unusable data. Moreover, regarding storage space, we next also propose which the most efficient way to store record data is.

4.2.3 Log File Normalization

Log data normalization or unification is gaining attention from the autonomic computing community [34] as a way to transform proprietary and heterogeneous formatted log data to a standard log data format.

In [28], the task of structuring event log data can be defined as the processes which provide structure to the semi-structured textual event log data and persist the resulting data structure for the later processing by analysis tools. Real e-Learning scenarios usually are formed by several LMS. Therefore, the input of the process is a set of LMS logs files generated by each source. As shown above, every log file, such as BSCW and UOC, has its own format showing strong differences in the formatting styles (e.g. in UOC Virtual Campus a log record is a text line in the text file whilst in BSCW each line represents an attribute value). Moreover, we cannot consider either unifying or normalizing those logs generated by the same Web Server (e.g., both Moodle and UOC Virtual Campus use Apache Web Server, but they log different information stored in different format); hence a preliminary process is needed in order to normalize these sources following an unified format. To this end, we propose the following tuple:

$$L = (u, t, a, [v] *)$$

which represents an user u performing an action a which occurs in time t. A list of values $[v] *$ is associated to the action. An example of a $(a, [v] *)$ instance could be:

$$(create_{document}, document.txt, 1024KB)$$

where first action-value is the filename of the document and the second is the size of the document.

Once we have normalized the log files, the resulting data structure persists for later data processing and analysis [28]. Next we proceed with a log data processing approach.

4.3 Parallel Processing Approach

The parallel implementation in the distributed infrastructures that we propose in this subsection follows the MapReduce paradigm [31]. Therefore, we introduce first our MapReduce model on the normalization of different LMS log files, namely BSCW and UOC, described in previous section. The results obtained will conduct our parallel implementation approach based on Hadoop and Cluster Computing presented in the next sections.

4.3.1 MapReduce Paradigm

We can assume that each log file type is a semi-structured text file with record-oriented structure, and the input data set is made up of a large number of files storing log information (e.g. each LMS, log per day, etc.). The input may be represented as:

$$I = \{Log_l^i\}, l \in L, i \in I$$

where *L* is the set of LMS, and *I* is the set of log files in a LMS.

The MapReduce paradigm works by splitting the processing into two stages, the map phase and the reduce phase, and each phase has key-value pairs as input and output. Therefore, we define the tasks in the Map phase and those processed in the Reduce phase, selecting the input and output keys for each phase. In this paradigm, the output from the map function could be processed by the framework before being sent to the reduce function.

The Map phase takes as input a record stored into a log file in I; the key of this record is the offset in the file. When the map function receives the record, it will be processed following the normalization

process which was presented above, and this output will be the input for the reduce function. At this point, we can decide among several alternatives dealing with reduce function. In order to only store normalized data, the reduce task does not perform additional work and store the output of map function in the distributed file system. In addition, the reduce function may be used to compute a relevant component as presented in the previous section. In that case, one of the keys is the student and the reduce function calculates the result of the parameter selected (e.g. number of documents created by the student, total session time, sum of ratings, etc.).

4.3.2 Hadoop

The abstract model proposed in above section supporting the MapReduce paradigm will be implemented in the parallel platform of Apache Hadoop[4]. In [31] MapReduce is presented as a model oriented to further implementations in Hadoop, hence we take this work as main reference in order to design our normalization LMS log files MapReduce framework.

Hadoop MapReduce job is defined as a unit of work that the client wants to be performed [31] consisting in: the input data, the MapReduce program, and configuration information. Then, Hadoop runs the job by dividing it into tasks of two types: map tasks and reduce tasks. There are also two types of nodes: job tracker, which coordinates the paralleling process, and several workers that perform the target work. Hadoop divides the input to a MapReduce job into fixed-size pieces and creates one map task for each split, which runs the map function for each record in the split. It is important to note that the number of reduce tasks is not ruled by the size of the input.

The implementation of map and reduce function is based on these previous works [28], [30], which deal with BSCW and UOC Virtual Campus log formats. Once the logs are computed by the event extractor functions, the output is normalized following the model presented.

4.3.3 Record Taxonomy and Implementation

The implementation of our parallel approach is in Java, which is compatible with Hadoop. Although certain implementation details are omitted, in this section we present the main services developed. These services are based on a study of the types of records registered in UOC Virtual Campus log files.

The core of the service is implemented in the Java class *Action*, which offers the main methods to process a record (i.e. a log file line). An action object represents something which has occurred in the Virtual Campus as described in UOC Virtual Campus Log Files. The main services and functions offered by the class Action are a set of get methods (e.g. get_date(), get_ip(), get_junk(), etc.) intended to parse the log line, the following classification summarizes the output records which can be managed using these methods:

- Record (R). Logs file line.
- Invalid Record (IR). An IR is a record which does not have a valid key. As previously stated, a valid key is a tuple with these components: session, IP and time.
- Valid Record (VR). In contrast, a VR contains each necessary field to form a valid record key.
- Request Record (RR). This type of record is a VR, which has requests and the server generates a 200 return code value. The set of CR includes Junk, Analysis and System records (which are defined below).
- Short Record (SR). If a VR does not reach conditions of RR (i.e. request and 200 code).
- Junk Record (JR). We define a JR as a RR on which we can ensure that does not contain relevant data. For instance, an image file requested by the client when creating a user web interface. In other words, a JR is valid but it does not contain useful data.
- Analysis Record (AR). Over the set of CR and those which are not JR, we select such records, which are relevant to a specific analysis. As we will present below, we select eleven representative

actions that a student may perform in the UOC Virtual Campus (e.g., an action may be a student accessing to a classroom environment).

- System Record (SR). Since the set of AR is selected for a specific case study, there exists a certain amount of Request Records that are not considered in the analysis. We name this type of records, System Records.

4.3.4 Type of Records

Of particular relevance is the amount of records computed of each type described above. These results will determine the best approach, sequential or parallel, to design the processing log model.

We run four types of records benchmarks for 10, 100, 1.000, and 10.000 MB log files. Results of these tests are shown below (see Fig. 2):

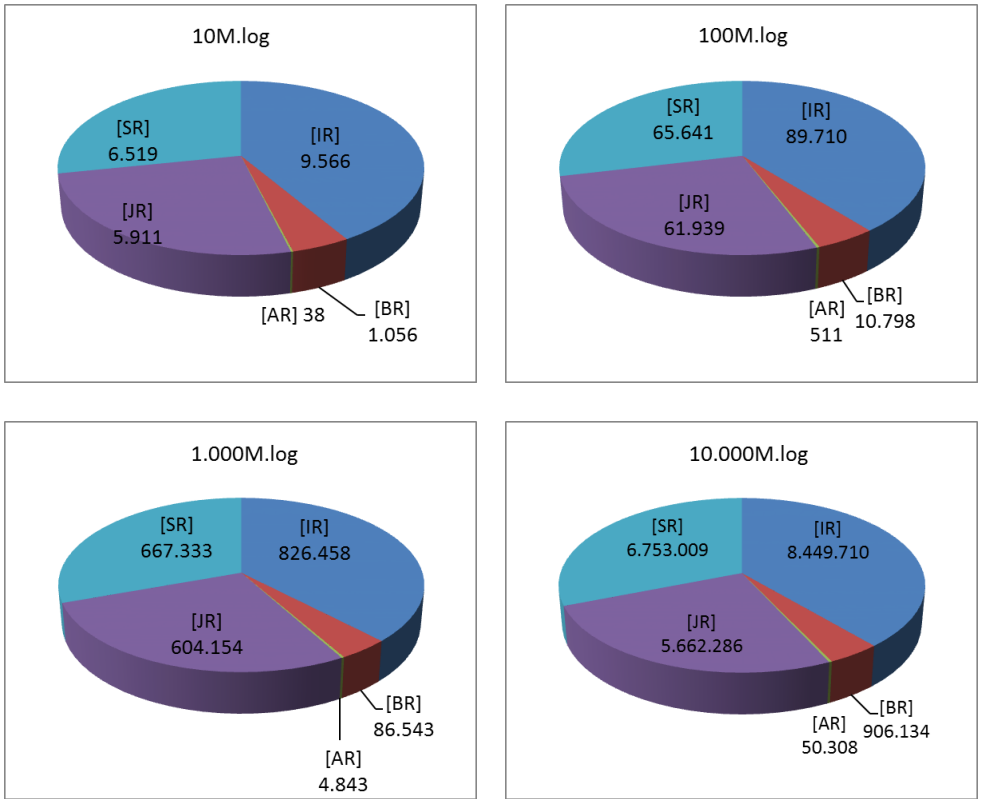

Figure 2. Types of records benchmarks.

As can be observed in Fig. 2, the amount of AR over the total data set is very low. Therefore, we need a pre-processing phase in order to extract useful information from logs files. Moreover, the average of each type of record is similar over the four tests; the size of the file does not generate different averages. Even when extending the study to more than the eleven selected actions in AR, the amount of BR is also very low.

4.3.5 *Results for Analysis Records*

We select those records that are relevant to our specific analysis associated to the actions performed in the LMS[1], which must be analyzed. The following table shows the name of each action, a short description, and the number of user actions computed in each input log file.

Table III: Trustworthiness Basic Indicators

Name	Description	Benchmark (xMB)										
		1	*10*	*50*	*250*	*100*	*500*	*1000*	*2000*	*4000*	*8000*	*10000*
Classroom	Access to a classroom environment	2	21	118	229	603	1112	2222	4429	8861	17624	22388
Login	Login LMS session	0	4	18	47	106	217	461	904	1717	3623	4617
Logout	Logout LMS session	1	2	3	8	17	23	67	155	325	747	954
File	Download a file	0	7	57	119	253	524	934	1982	4008	7841	9958
Mail	Load the E-Mail service	0	0	1	3	11	37	118	232	466	835	973
Community	Community campus	0	0	3	9	22	55	123	263	536	1125	1390
Services	General services	0	0	1	7	19	38	66	126	273	521	642
Secretary	Secretary´s office service	0	2	13	33	69	127	295	648	1283	2563	3081
Profile	Load an user profile	0	1	7	12	22	40	69	127	235	472	592
News	UOC news service	0	1	7	14	28	41	78	184	431	901	1087
Help	Help Desk	0	0	1	2	6	11	20	49	127	264	316

4.4 *Hadoop Processing Logs Implementation*

In this section, we present the most significant aspects of deploying and implementing a MapReduce paradigm intended to manage log data as described in this paper.

4.4.1 *MapReduce Java Implementation*

MapReduce Java Implementations are completely based on the class *Action*, which was early developed to test sequential results with regarding to time and records types benchmarks. We have developed two separated Java applications, which are presented in this section.

UOCLogDriverClean program normalizes UOC logs files cleaning unnecessary data. Only records in the AR set are considered as outcome, and the other record sets are ignored. In order to improve computational performance, the algorithm progressively inspects each condition in a well-arranged way, that is, firstly the most restrictive and general condition (e.g. *has_session*) and finally the most specific one (e.g. *has_opa*). The mapper receives as parameters a pair (key, value), where the key is automatically generated by Hadoop and the value is a line of a log file. The output is a different pair where the key is the record id and the value is the request. It is important to note that, in this case, we do not use Reduce function because we are not running Reduce tasks (i.e. grouping, computing, etc.).

UOCLogDriverCountOp is the second application developed and it has both Map and Reduce functions. In this case, our goal is computing aggregate data by computing the sum of each action type (the same outcome as in sequential implementation). The *UOCLogDriverCountOp* Map code is similar to *UOCLogDriverClean* implementation, however, output key-value for the type of value has been denied

[1] Although we focus on students' e-Learning and behavior actions, additional technological information, such as students' device or IP control, could be also included in the study.

as integer to compute each instance. When Map function has generated each key-value pair, the output is combined over the value key and processed by the Reduce function.

Finally, collected data are stored in the output directory, which is defined when the job is executed.

4.5 Analysis of the Results

Once the MapReduce applications have been developed, before running the jobs in parallel processing, network and distributed file systems are needed. Hadoop Distributed File System (HDFS) supports large datasets across multiple hosts to achieve parallel processing. HDFS is a block-structured file system based on splitting input data into small blocks of fixed size, which are delivered to each node in the cluster. We use HDFS as Hadoop MapReduce storage solution; therefore some file system configuration tasks are needed, such as create user home file and define suitable owner, create MapReduce jobs input directory, upload log files and retrieve results actions.

In Table IV and the corresponding Fig. 3, we can see comparative results of the battery of tests with multiple Hadoop nodes (i.e. 2, 4, 6, 8 and 10 workers) in RDlab[5] cluster. Note that 0-node shows results of local sequential processing benchmark. Furthermore, we have carried out additional file system integration processes by running Hadoop jobs over the open-source Lustre[6] file system, which is deployed in the RDlab.

From this experimental study, we can see that the results do not grow linearly anymore. We can also see that by using a distributed MapReduce Hadoop infrastructure, a considerable speed up is achieved in processing large log file data as shown in Table IV last column (i.e. more than 50% for infrastructures with more than 4 nodes and more than 75% for 10 nodes). Regarding log file size, for too small values, the overhead introduced by the MapReduce framework, when sending the parts to the nodes and combining output data is noticeable and the framework control tasks, spends too much time managing and distributing data. On the other side, values of the task size close to 3,000 MB considerably diminish this amount of time in comparison with the total processing time. Moreover, Reduce tasks spend too much time when the number of nodes is low.

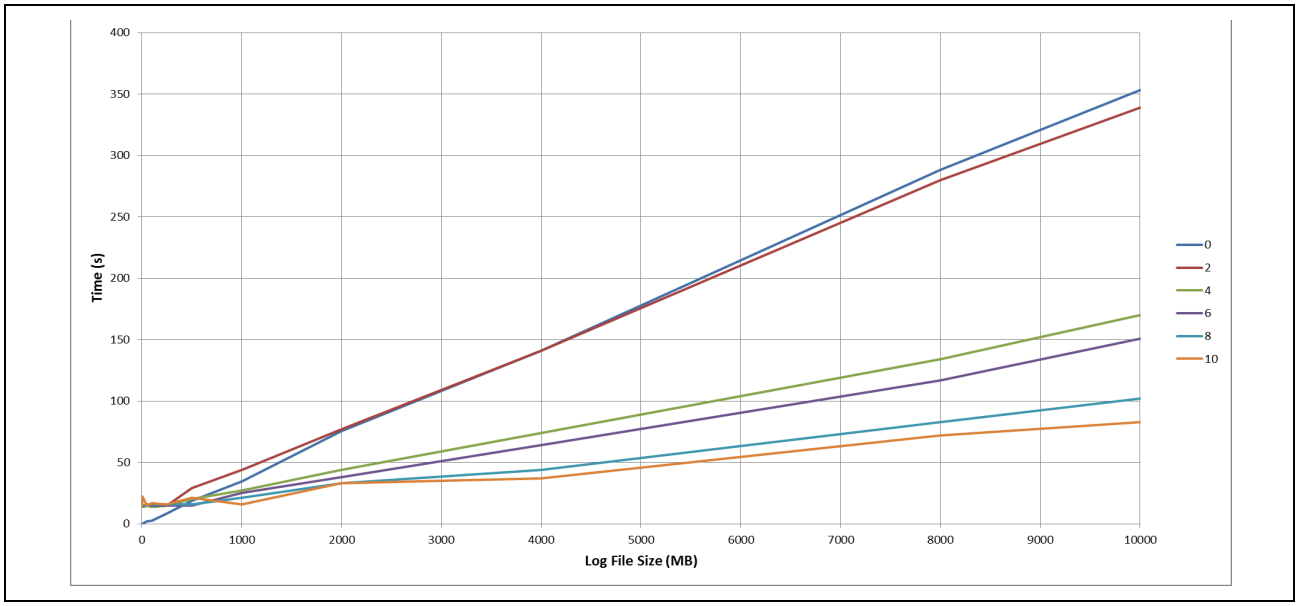

Figure 3. Comparative Mapreduce Results.

Table IV: Comparative MapReduce Results.

Nodes	Log File Size											
	1	*10*	*50*	*250*	*100*	*500*	*1000*	*2000*	*4000*	*8000*	*10000*	*Speed up (%)*
0	0	0	2	2	9	19	35	75	141	288	353	
2	14	14	15	14	15	29	44	77	141	280	339	**4%**
4	15	15	15	14	15	20	27	44	74	134	170	**52%**
6	14	14	16	15	15	15	25	38	64	117	151	**57%**
8	16	14	15	16	16	16	21	33	44	83	102	**71%**
10	14	22	15	17	16	21	16	33	37	72	83	**76%**

5 Conclusions and Further Work

In this paper, we first motivated the need to improve information security in e-Learning and in particular in CSCL activities. Then, we proposed a methodological approach to build a security model for CSCL activities with the aim to enhance standard technological security solutions with trustworthiness factors and rules. As a result, the guidelines of a holistic security model in on-line collaborative learning through an effective trustworthiness approach were first proposed. However, as learners' trustworthiness analysis involves dealing with large amount of data generated along learning activities, processing this information is computationally costly, especially if required in real-time. To this end, and as a main contribution of this paper, we proposed a parallel processing approach that can considerably decrease the time of data processing, thus allowing for building relevant trustworthiness models to support learning activities even in real-time.

The implementation of our parallel approach faced two important challenges: handle several formats of logs files coming from different LMS and the large size of these log files. We showed how to normalize different log file structures as an input for the MapReduce paradigm to manage huge amounts of log data in order to extract the trustworthiness information defined in our model.

Finally, we used distributed infrastructure, such as Hadoop and Cluster Computing, to implement and evaluate our parallelization approach for massive processing of log data. The experimental results showed the feasibility of coping with the problem of structuring and processing ill-formatted, heterogeneous, large log files to extract information on trustworthiness indicators and levels from learning groups and ultimately fill a global framework devoted to improve information security in e-Learning in real-time. We eventually conclude that it is viable to enhance security in CSCL activities by our trustworthiness model, though taking on the overhead caused by the use of distributed infrastructure for massive data processing.

As ongoing work, we plan to improve the MapReduce configuration strategies that would result in improvement of a parallel speed-up, such as customized size of partitions. Furthermore, we are investigating normalized trustworthiness improvements to extend the model presented in this paper.

6 References

[1] T. Koschmann, "Paradigm Shifts and Instructional Technology," in *CSCL: Theory and Practice of an Emerging Paradigm*, T. Koschmann, Ed. Mahwah, New Jersey: Lawrence Erlbaum Associates, 1996, pp. 1–23.

[2] P. Dillenbourg, "What do you mean by collaborative learning?," in *Collaborative-learning: Cognitive and Computational Approaches*, P. Dillenbourg, Ed. Oxford, UK: Elsevier Science, 1999, pp. 1–19.

[3] E. R. Weippl, *Security in e-learning*. New York, NY: Springer, 2005.

[4] C. J. Eibl, "Discussion of Information Security in E-Learning," Universität Siegen, Siegen, Germany, 2010.

[5] G. Kambourakis, D.-P. N. Kontoni, A. Rouskas, and S. Gritzalis, "A PKI approach for deploying modern secure distributed e-learning and m-learning environments," *Computers & Education*, vol. 48, no. 1, pp. 1–16, 2007.

[6] Trustwave, “Trustwave 2012 Global Security Report,” Trustwave, 2012.

[7] CSO Magazine, US Secret Service, Software Engineering Insistute CERT Program at Carnegie Mellon University, and Deloitte, “2011 Cybersecurity Watch Survey,” CSO Magazine, 2011.

[8] S. Narkhede and T. Baraskar, “HMR Log Analyzer: Analyze Web Application Logs Over Hadoop MapReduce,” *International Journal of UbiComp*, vol. 4, no. 3, pp. 41–51, Jul. 2013.

[9] V. Ciesielski and A. Lalani, “Data mining of web access logs from an academic web site,” in *Design and Application of Hybrid Intelligent Systems*, A. Abraham, M. Köppen, and K. Franke, Eds. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2003, pp. 1034–1043.

[10] S. Caballé, C. Paniagua, F. Xhafa, and T. Daradoumis, “A Grid-Aware Implementation for Providing Effective Feedback to On-Line Learning Groups,” in *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, vol. 3762, R. Meersman, Z. Tari, and P. Herrero, Eds. Springer Berlin Heidelberg, 2005, pp. 274–283.

[11] K. Borcea, H. Donker, E. Franz, A. Pfitzmann, and H. Wahrig, “Privacy-Aware eLearning: Why and How,” presented at the World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA) 2005, Chesapeake, Virginia, 2005, pp. 1466–1472.

[12] C. J. Eibl, “Privacy and Confidentiality in E-Learning Systems,” presented at the Fourth International Conference on Internet and Web Applications and Services, 2009 - ICIW 09, 2009, pp. 638 –642.

[13] S. K. Ferencz and C. W. Goldsmith, “Privacy issues in a virtual learning environment,” in *Cause/Effect, A practitioner’s journal about managing and using information resources on college and university campuses*, vol. 21, Educause, 1998, pp. 5–11.

[14] E. R. Weippl, “Security in E-Learning,” in *Handbook of information security Vol. 1, Key concepts, infrastructure, standards and protocols.*, vol. 1, 3 vols., Hoboken, NJ: John Wiley & Sons, Inc., 2006, pp. 279–294.

[15] J. Miguel, S. Caballé, and J. Prieto, “Providing Security to Computer-Supported Collaborative Learning Systems: An Overview,” presented at the Fourth IEEE International Conference on Intelligent Networking and Collaborative Systems (INCOS 2012), Bucharest, Romania, 2012, pp. 97–104.

[16] J. Miguel, S. Caballé, and J. Prieto, “Security in Learning Management Systems: Designing collaborative learning activities in secure information systems,” *eLearning Papers. European Comission: elearningeuropa.info*, 2012.

[17] J. Miguel, S. Caballé, and J. Prieto, “Information Security in Support for Mobile Collaborative Learning,” presented at the The 7th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2013), Taichung, Taiwan, 2013.

[18] J. Miguel, S. Caballé, and J. Prieto, “Providing Information Security to MOOC: Towards effective student authentication,” presented at the 5-th International Conference on Intelligent Networking and Collaborative Systems INCoS-2013, Xian, China, 2013, pp. 289 – 292.

[19] J. D. Demott, A. Sotirov, and J. Long, *Gray Hat Hacking, Third Edition Reviews*, 3rd ed. New York: McGraw-Hill Companies, 2011.

[20] Z. Wu, Y. Ou, and Y. Liu, “A Taxonomy of Network and Computer Attacks Based on Responses,” in *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on*, 2011, vol. 1, pp. 26 –29.

[21] L. Pappano, “The Year of the Mooc,” *The New York Times*, 2012. .

[22] B. Schneier, “The psychology of security,” in *Proceedings of the Cryptology in Africa 1st international conference on Progress in cryptology*, Berlin, Heidelberg, 2008, pp. 50–79.

[23] C. Dai, D. Lin, E. Bertino, and M. Kantarcioglu, “An Approach to Evaluate Data Trustworthiness Based on Data Provenance,” in *Secure Data Management*, vol. 5159, W. Jonker and M. Petković, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 82–98.

[24] P. Bernthal, “A survey of trust in the workplace,” HR Benchmark Group, Pittsburg, PA, Executive Summary, 1997.

[25] A. Abdul-Rahman and S. Hailes, “Using Recommendations for Managing Trust in Distributed Systems,” in *Proceedings of the IEEE Intl. Conference on Communication, Malaysia*, 1997.

[26] I. Ray and S. Chakraborty, “A Vector Model of Trust for Developing Trustworthy Systems,” in *Computer Security – ESORICS 2004*, vol. 3193, P. Samarati, P. Ryan, D. Gollmann, and R. Molva, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 260–275.

[27] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Toward Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness,” *ACM Trans. Internet Technol.*, vol. 7, no. 4, Oct. 2007.

[28] F. Xhafa, C. Paniagua, L. Barolli, and S. Caballe, "A Parallel Grid-Based Implementation for Real-Time Processing of Event Log Data of Collaborative Applications," *Int. J. Web Grid Serv.*, vol. 6, no. 2, pp. 124–140, Jun. 2010.

[29] S. Caballe, F. Xhafa, and T. Daradoumis, "A Grid Approach to Efficiently Embed Information and Knowledge about Group Activity into Collaborative Learning Applications," in *The Learning Grid Handbook*, vol. 2, Amsterdam, The Netherlands: IOS Press, 2008, pp. 173–197.

[30] S. Caballé and F. Xhafa, "Distributed-based massive processing of activity logs for efficient user modeling in a Virtual Campus," *Cluster Computing*, Apr. 2013.

[31] T. White, *Hadoop: the definitive guide*, Third edition. Beijing: O'Reilly, 2012.

[32] W. Appelt, "What groupware functionality do users really use? Analysis of the usage of the BSCW system," pp. 337–341.

[33] The Apache Software Foundation, "Apache HTTP Server Version 2.2 Documentation," 2014. [Online]. Available: http://httpd.apache.org/docs/2.2/.

[34] F. Salfner, S. Tschirpke, and M. Malek, "Comprehensive logfiles for autonomic systems," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 211.

---

[1] The Open University of Catalonia is located in Barcelona, Spain. The UOC offers distance education through the Internet since 1994. Currently, about 60,000 students and 3,700 lecturers are involved in 8,300 online classrooms from about 100 graduate, post-graduate and doctorate programs in a wide range of academic disciplines. http://www.uoc.edu

[2] https://moodle.org/mod/forum/view.php?f=996

[3] http://httpd.apache.org/

[4] http://hadoop.apache.org

[5] http://rdlab.lsi.upc.edu

[6] http://lustre.opensfs.org/