The ERP AuditLab - A prototypical Framework for Evaluating Enterprise Resource Planning System Assurance

Nick Gehrke
Hamburg University, Germany
nick.gehrke@wiso.uni-hamburg.de

Abstract

Enterprise Resource Planning Systems (ERP) are comprehensive systems used in global organizations for distributed and complex transactions. Organizations are strongly dependent on these systems since they control critical business process flows. Assurance for these systems is needed to guarantee proper functionality, being in line with compliance requirements and to prevent fraudulent activities. We present a software prototype called the ERP AuditLab for auditing so called application controls in ERP systems. A case study is presented to give an impression about important application controls. We use the case study to derive the requirements for the ERP AuditLab.

1. Introduction

Auditing Enterprise Resource Planning Systems (ERP systems) is a very complex task since it requires deep technical knowledge of a systems’ customizing and also knowledge about the relevance of settings for the internal control system. In an ERP environment auditing becomes more complex [21] and internal controls more difficult to assess for auditors [14] [15] [17]. In practice, (IT)-auditors are specialized to a specific ERP-system and the quality of an audit strongly depends on the experience of the auditor with the system(s) in scope. Since the customizing of a system could be queried automatically and “only” needs to be interpreted through an auditors’ perspective, we develop a software framework called the ERP AuditLab to provide a useful and flexible IT-artifact considered to be a computer assisted audit technique (CAAT) to auditors for system reviews. One of the main requirements of the ERP AuditLab is that additional control tests can be implemented without writing any new source code. The requirements for the ERP AuditLab have been derived from a real-life project which will be presented as a case study.

The paper is structured as follows. In the next section we provide a brief overview about related work and existing research. After that definitions of CAAT and internal control(s) are provided. Additionally, the scope of the ERP AuditLab is pointed out. We then present the case study and derive the requirements for the ERP AuditLab. Following, a software prototype is presented and finally evaluated.

2. Related Work

We conducted a literature investigation about audit automation since the ERP AuditLab also aims to automate auditing. Modeling and querying an internal control system with TICOM has been initially described and implemented by Baily [3] in the literature. TICOM provides an internal controls description language (ICDL) and is focused on internal controls modeling whereas our approach focuses less on modeling and constitutes more a real-time internal control scanning approach for ERP systems as a useful IT artifact and aid for internal and external auditors. Chang and Wu [7] develop a Computer Auditing System for the oracle ERP system. They examine so called control items in the oracle ERP system. Their approach differs from our ERP AuditLab since we wanted to design software independent of a specific ERP system. Other authors developed approaches to automate the creation of audit reports [22].

Another area of research linked to our research is “continuous auditing”. Continuous auditing can be defined as "a methodology that enables independent auditors to provide written assurance on a subject matter using a series of auditors' reports issued simultaneously with, or a short period of time after, the occurrence of events underlying the subject matter." ([8], an overview about the subject gives Coderre [9], a literature overview gives Brown et al. [6] and Murcia et al. [19]). Continuous auditing tries to establish a real-time auditing and reporting approach and introduces many ideas for automation of audits [4]. Many authors provide concepts how to automate audits.
and evaluate compliance (Hasan and Stiller [11] provide a generic model). Different approaches as modeling continuous auditing with security patterns [16], triggering events [13], implementations with SOA and web services [23] [24] or expert systems [1] have been subject to discussion. Additionally, real implementations of continuous auditing systems [2] and a survey about the relevance of continuous auditing [20] can be found in the literature.

Continuous auditing gives a good clue about automation of auditing tasks but we think that continuous auditing cannot be compared with our approach due to one crucial reason. Continuous audit systems are embedded in a companies’ stable environment whereas external financial auditors are independent and need their own flexible programs and tools under their own control. There is little literature about this integration aspect of a companies’ and the auditors’ system [18].

3. Definitions

3.1. Computer Assisted Auditing Techniques

From a broader perspective, Computer Assisted Auditing Techniques or Tools (CAAT) include “any use of technology to assist in the completion of an audit. This (broad) definition would include automated working papers and traditional word processing applications ...” [5]. According to a more narrow definition which is used in the paper on hand, CAATs are “computer tools that extract and analyze data from computer applications” [5]. In this paper, the analysis is focused on application controls (see below).

3.2. Internal Control

Internal controls are a part of the internal control system used by companies and other organizations in order to comply with internal and external governance requirements. Traditionally, internal controls are an indirect instrument to assure a true and fair view representation of financial statements. The Sarbanes Oxley Act (SOX), issued in 2002 as a reaction of breakdowns of big companies like Enron and WorldCom, emphasizes the importance of the internal control system. The internal control system and thus the audit of business processes became a separated subject of auditing (in addition to the balance sheet) since external auditors have to report on the adequacy of the company’s internal control over financial reporting (ICFR).

The Committee of Sponsoring Organizations of the Treadway Commission (COSO) “defines internal control as a process, effected by an entity’s board of directors, management and other personnel. This process is designed to provide reasonable assurance regarding the achievement of objectives in effectiveness and efficiency of operations, reliability of financial reporting, and compliance with applicable laws and regulations.” [10] [12, p. 663].

Internal controls can thus be understood as activities for achieving internal control about processes. Although the COSO describes that internal control is effected by persons, internal control activities itself can be embedded in systems and operate automatically.

3.3. Taxonomy of Internal Controls

The scope of internal controls is not limited to financial statement assurance and can be focused on other control objectives. Internal controls have diverse properties. The following table tries to summarize the most important properties of internal controls in order to categorize the “universe of internal controls”.

<table>
<thead>
<tr>
<th>Property</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature</td>
<td>Manual Automated</td>
</tr>
<tr>
<td>Frequency</td>
<td>Continuous Daily Weekly Monthly Quarterly Other</td>
</tr>
<tr>
<td>Assertion</td>
<td>Completeness Accuracy Validity Access Cut Off Other</td>
</tr>
<tr>
<td>Importance</td>
<td>Key Standard</td>
</tr>
<tr>
<td>Scope</td>
<td>Company Entity Process Activity Transaction</td>
</tr>
<tr>
<td>Timeliness</td>
<td>Preventative Detective</td>
</tr>
<tr>
<td>Subject</td>
<td>Physical Asset Fin. Asset Data Asset Human Asset</td>
</tr>
<tr>
<td>Objective</td>
<td>Reporting Security Compliance</td>
</tr>
<tr>
<td>Documentation</td>
<td>Documented Informal</td>
</tr>
<tr>
<td>Integration</td>
<td>Process integr. Process Independent</td>
</tr>
<tr>
<td>Responsibility</td>
<td>Top-Mgmt. Busin.-Mgmt. IT-Mgmt.</td>
</tr>
</tbody>
</table>

Table 1: Properties of Internal Controls

Table 1 shows the properties of internal controls. For the ERP AuditLab prototype, the application controls are the main focus. Application controls are inherently embedded in software or can be switched on with customizing. Examples are automatic reconciliation procedures, prevention of entering duplicate transactions or system forced approvals and 4-eyes principles. Application controls are efficient as well as effective. Efficiency is achieved since these kind of controls are automatic and do not involve any manual activities. Effectiveness is achieved since customizing
only needs to be adjusted once. Table 1 also shows the usual properties of application controls (bold entries). These are normally automated, they continuously work in the IT system, operate on the process, activity or transaction level, protect financial or data/information assets, are well documented since they need to be switched on in the customizing and business or IT management takes over responsibility.

4. Case Study

This case study resulted from a real-life project conducted in one of the biggest companies in Germany. The aim of the project was to develop and implement an application controls compliance frame for SAP R/3 systems.

4.1. Scope of the Case Study

The company maintained a few hundreds of SAP R/3 installations but not all were used in the investigation. The business processes in scope were the SAP basis administration process and the purchase-to-pay process. The aim of the project was to identify application controls which can be assessed automatically without any manual activities and without taking the specific local process environment into account. The outcome of testing of the application controls could only be ‘switched on’ or ‘switched off’. The identified application controls can be considered to be only a basic application compliance framework since other necessary complex application controls need a manual interpretation but have not been taken in consideration. About 33 control tests have been identified and implemented in the SAP ABAP programming language. These ABAP programs have been manually transported in the systems in scope. Results are transmitted to a central compliance dashboard for reporting reasons.

4.2. Results of the Case Study

After the programs have been transported into the SAP systems of the company first results could be gained. To give an impression about the nature of application controls a part of the implemented application controls are listed below.

Basis area:
- ‘logging of users with extensive authorizations needs to switched on’
- ‘Enable logging of changes to critical tables’
- ‘Use of strong password and login parameters is required’

Purchase-to-pay area:
- ‘Prevent posting of duplicate invoices’
- ‘Enable customizing parameters for three-way-match (purchase order and invoice and goods receipt)’
- ‘Require approval for changes of sensitive master data fields’
- ‘Prevent payments to alternate payee’
- ‘Enable duplicate vendor check’
- ‘Define mandatory fields during vendor master data maintenance’
- ‘Prevent use of one-time vendor accounts’

In the figure columns show SAP systems from different business areas. Only the first 7 application controls are shown. Overall, 63 SAP systems have been investigated. The switch-on-rate of the 33 application controls turned out to be around 60%. System and business owners needed to increase the rate and switch on additional application controls. The number in brackets shows how many systems switched on the control. A green flag means that all systems switched on the control, a red flag means that no system switched on the control.
4.2. Derived Requirements

Based on the case study above we derived basis requirements for the ERP AuditLab as follows:

(1) **Independent Software.** As explained, the application control tests in the case study are implemented within the SAP systems and need to be costly transferred to each system instance. The ERP AuditLab should be less system integrated and under full control of the auditor to be a flexible and useful IT-artifact. The ERP AuditLab should be more a sophisticated Computer Assisted Audit Tool (CAAT) than a continuous audit environment.

(2) **Scalable and Extensible Software.** Since the control tests in the case study are individually programmed in the ABAP programming language, additional tests to be implemented need additional individual programming and new transports and distribution to the SAP systems. The ERP AuditLab should be only a framework where new control tests can be constructed without implementing any source code.

(3) **Central Result Database.** In order to have an overview about the status of several different systems, audit results should be able to be consolidated on a central server instance.

5. Software Prototype

According to the case study and the derived requirements we developed a software prototype called the ERP AuditLab. The software prototype completely operates outside the ERP system to be examined. Data to be evaluated needs to be downloaded. The overall examination or audit process is shown in figure 2.

Each phase will be detailed in the next sections.

5.1. Functional Structure

Before we start with details figure 3 shows the overall functional structure of the framework.

Controls and control tests need to be categorized in a useful hierarchy. This hierarchy is shown in the boxes from ‘model’ to ‘sub process’. A model could be ‘Standard Business Processes’ or specific branch models like ‘Business Processes for TelCo Providers’. Streams are general processes like ‘purchase-to-pay (ptp)’ or ‘order-to-cash (otc)’. Processes belong to streams. ‘Invoice Processing’ could be an example for a process within the stream ‘ptp’. Sub processes further detail processes e.g. ‘Invoice processing without purchase order’ could be an example. Within sub processes, process risks are inherent e.g. ‘Vendor master data could be manipulated leading to fraudulent payments’. Control Objectives aim to compensate these risks. Abstract controls are system related controls without focusing on a specific ERP system, e.g. ‘The system incorporates a control to prevent processing duplicate vendor invoices’. The construct ‘Abstract Control’ emphasizes the approach as being in general independent on a specific ERP system. Initially the ERP Control specifies the abstract control for a specific ERP system e.g. ‘The parameter x is switched on to prevent processing duplicate vendor invoices.’ All elements discussed so far only categorize or document controls within a system. The next element, the Control Test contains analysis logic and audit knowledge to examine the system in scope. A Control Test contains an arbitrary number of Test Rules. These Test Rules can be arbitrary combined described by Rule Dependencies. Each automatic Control Test leads to three different predefined ‘answers’ formulated automatically in a textual style which should be also understood by non-technical oriented persons: (1) the
result of the audit of the control, (2) an interpretation and assessment of the result and (3) a recommendation for remediation if necessary.

5.2. Data Acquisition and Preparation

Data acquisition is the first step to evaluate assurance of an ERP system. The data extraction process within a system audit is often more complicated than presumed. Direct database connection with mechanisms like ODBC (open database connectivity) might be not possible since the IT department of the client tends to seal off IT applications for direct database access for security reasons. Often the auditor has a read-only user account in the system and extracts reports or files manually. The following data extraction mechanisms do exist:

1. Direct database access e.g. with ODBC.
2. Table extraction using the application (e.g. with transaction SE16 in a SAP system).
3. Report extraction using the normal reporting functionalities.
4. Remote data extraction e.g. with the Remote Function Call (RFC) mechanism in a SAP system.

Our ERP AuditLab prototype implements opportunity (2) and (4) since these extraction mechanisms guarantee structured data lines. Reports (3) might be structured hierarchical for a human’s eye convenience but also heavy to read for a machine. An implementation of (3) is planned to follow. Direct database access (1) is trivial.

Having acquired the data, tables to be examined are stored in a local ERP AuditLab database. At this stage, fields of all extracted tables are of type text. The next step is to prepare and aggregate tables for controls testing.

For data preparation four constructs are needed: (1) Data Objects, (2) Data Object Definitions, (3) Data Object Dependencies and (4) SQL Statements for Data Objects.

Data Objects contain all tables or data objects needed for an audit. These objects are not only the tables acquired from the system (the original tables), but also tables which filter or aggregate data since the control test analyses do normally not operate on original tables.

Data Object Definitions contain the data types of the fields of a Data Object.

Data Object Dependencies define how a data object is to be constructed. Dependencies only define a target data object and an arbitrary number of source data objects. Consider the following entries in the Data Object Dependencies:

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Target Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
</tr>
</tbody>
</table>

Figure 4 shows how data objects could depend on each other. Obviously, Data Objects of arbitrary complexity can be constructed with a simple Object Dependency list. If a control test needs e.g. Object E then the object will be constructed according to the Object Dependencies. This is a recursive process. Constructing Object E will call the construction method for D and C. C does not need to be constructed since it is an original table. The construction of D will invoke recursively the construction of A and B which are again original tables and thus the overall construction of E terminates. Please note that Object Dependencies only describe what object depends on what other object but do not define how to do the construction. For this purpose all non-original Data Objects (in figure 3 the objects D and E) need a construction rule.

In our prototype this construction rule is simply a SQL-Statement mapped to the objects D and E. Using these constructs an arbitrary number of Data Objects with arbitrary complexity can be defined without extending or writing additional source code (except SQL-statements to be inserted in a database table). Thus, Data Objects can fully be implemented within the customizing of the prototype.

5.3. Automatic Control Tests

To evaluate the status of application controls, the Data Objects need to be examined. A Data Object is assigned to each Control Test. If the Control Test is performed, the assigned Data Object is constructed as described in figure 4. The logic or knowledge how to examine the assigned Data Object can also be defined in the customizing of the ERP AuditLab. No additional
source code is needed if new Control Tests are to be implemented. To perform the Control Test, an arbitrary number of simple rules can be defined. These simple rules always only result in TRUE or FALSE.

Table 2: Simple Rules

<table>
<thead>
<tr>
<th>Test-ID</th>
<th>Variable</th>
<th>Data-objectID</th>
<th>Datafield</th>
<th>Operator</th>
<th>ParameterID1</th>
<th>ParameterID2</th>
</tr>
</thead>
<tbody>
<tr>
<td>9x</td>
<td>E</td>
<td>RSAU</td>
<td>=</td>
<td>RSAUVALUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9t</td>
<td>E</td>
<td>CURRPROF</td>
<td>&gt;</td>
<td>SAPALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9u</td>
<td>E</td>
<td>SIZE</td>
<td>RANGE</td>
<td>LOWERLEVEL</td>
<td>UPPERLEVEL</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 shows 3 example rules for a Control Test. Three data fields of Data Object E are evaluated. These data fields are compared against to-be-values. The to-be-values are not presented directly in the rule but are put behind placeholders or parameters (ParameterID1 and ParameterID2). The advantage is that placeholders or parameters can be bundled in different parameter profiles and thus an audit can be lackadaisical or rigorous depending on the used parameter profile for the to-be-values. The result (TRUE or FALSE) of each rule will be stored in a Variable (see correspondent column in table 2).

These simple rules can be combined with each other in an arbitrary complexity using the Rule Dependencies.

Table 3: Rule Dependencies

<table>
<thead>
<tr>
<th>Test-ID</th>
<th>Order</th>
<th>Else</th>
<th>Result</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 1</td>
<td>NO</td>
<td>ELSE</td>
<td>TRUE</td>
<td>(not(#t#))</td>
</tr>
<tr>
<td>9 2</td>
<td>NO</td>
<td>ELSE</td>
<td>TRUE</td>
<td>AND(AND(#t#,AND(#u#,not(#v#))),AND(AND(#w#,not(#x#)),AND(#y#,not(#z#))))</td>
</tr>
<tr>
<td>9 3</td>
<td>NO</td>
<td>ELSE</td>
<td>TRUE</td>
<td>AND(AND(#t#,OR(#u#,not(#v#))),AND(AND(#y#,not(#x#)),AND(#z#,not(#w#))))</td>
</tr>
<tr>
<td>9 4</td>
<td>NO</td>
<td>ELSE</td>
<td>TRUE</td>
<td>AND(AND(#t#,OR(#u#,not(#v#))),AND(AND(#x#,not(#w#)),AND(#z#,not(#y#))))</td>
</tr>
<tr>
<td>9 5</td>
<td>NO</td>
<td>ELSE</td>
<td>TRUE</td>
<td>AND(AND(#t#,OR(#u#,not(#v#))),AND(AND(#x#,not(#w#)),AND(#z#,not(#y#))))</td>
</tr>
<tr>
<td>9 6</td>
<td>YES</td>
<td></td>
<td>TRUE</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3 shows how simple rules can be combined to rules of higher complexity. Please note that variables defined in the simple rule set appear like #x#. The row ‘order’ controls in which order rules are processed. If no complex rule matches the entry with ‘yes’ in the column ‘else’ is chosen. The matching complex rule controls the outcome of the Control Test as shown in column ‘result’ in Table 3. Again, Control Test logic does not need any additional source code. New Control Tests can be easily transferred by copying database entries.

5.4. Inference and Recommendations

The above section demonstrated how Control Tests are performed. The next task is to evaluate (1) the result of the audit of the control, (2) an interpretation and assessment of the result and (3) a recommendation for remediation if necessary.

The result is determined by the logic of the Control Test (see table 3, column ‘Result’; table 4 column ‘No’). For each Control Test a predefined number of textual results need to be defined. Since textual results cannot always be completely static, placeholders can be put in the textual results.

Table 4: Textual Results

<table>
<thead>
<tr>
<th>Test-ID</th>
<th>TextualResult</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>There are totally #PLH_SAPALLUSER# users with extensive authorizations. #PLH_NOTLOGSAPALL# of them are not logged. Both of the standard users DDIC and SAP* are logged.</td>
</tr>
<tr>
<td>9</td>
<td>There are totally #PLH_SAPALLUSER# users with extensive authorizations. #PLH_NOTLOGSAPALL# of them are not logged. The standard user DDIC is not logged, but SAP* is.</td>
</tr>
</tbody>
</table>

Table 4 shows textual results for a Control Test. In this case two – slightly different – textual results have been defined. Please note that placeholders in the textual result are used to present exact results of the Control Test (placeholder in the example is #PLH_SAPALLUSER#). The placeholder will be replaced when the Control Test is performed.

An assessment of the result is technically done analogously to the evaluation of the textual result. Again, textual assessments need to be defined for each Control Test.

Table 5: Internal Control Assessment

<table>
<thead>
<tr>
<th>Test-ID</th>
<th>AssessmentTitle</th>
<th>AssessmentDescription</th>
<th>TypeOfAssessment</th>
<th>Assessment</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 1</td>
<td>Setting is appropriate</td>
<td>Logging of standard/anonymous users and users with extensive authorizations is activated in the system profile parameters.</td>
<td>Finding</td>
<td>Green</td>
<td>2</td>
</tr>
<tr>
<td>9 2</td>
<td>Setting is inappropriate</td>
<td>Logging of standard/anonymous users and users with extensive authorizations is deactivated in the system profile parameters.</td>
<td>Finding</td>
<td>Red</td>
<td>0</td>
</tr>
</tbody>
</table>

For each result alternative (column ‘No’ in table 4 and 5) an assessment alternative needs to be defined. In this case only two alternative assessments have been implemented. Normally, at least two alternatives need to be defined (one for a satisfying result and one for a bad result). Additionally, it can be defined if the result
should be interpreted to be a finding and the severity of the finding (Indicated e.g. as green/yellow/red).

The technical implementation of the recommendation according to a finding is analogous to the evaluation of the result and the assessment.

In addition to the textual assessment, a score can be assigned to a specific assessment alternative. The score can be used to calculate a total score for all control tests conducted or for the complete internal control system. The total score can be used to benchmark one organization with another or to compare one organization at different times.

Please note that an auditor should never blindly trust the automatic assessment of the ERP AuditLab. For this reason, the ERP AuditLab always demands a review by the auditor for all automatically derived results, assessments and recommendations (see figure below). If needed, the auditor can manually adjust the textual results, assessments or recommendations. Please also note that there can be Control Tests which cannot be assessed automatically since further issues need to be taken into consideration. In this case, assessments and recommendations do not need to be defined and the system only constructs Data Objects to ease the manual application of professional judgment of the auditor.

![Figure 5: Auditors’ review of automatically evaluated findings in the prototypical software framework](image)

### 5.5. Reporting and further Features

Since the described framework is highly structured, an automated reporting is straightforward. Because all results and textual descriptions are stored in separate data fields, SQL-Queries can be constructed and used in a report generator component.

We implemented the following reports which normally appear in the audit area:

1. **A testing status list** which only shows the title of the control test and the result (‘finding’ or ‘no finding’ respective status is ‘in-progress’ or ‘done’). This report can be used to get a quick status of the audit for audit managers.

2. **A test plan** with detailed description of the test result. This report shows all control test activities and the test results, assessments and recommendations. This report can be used for the detailed documentation of audit activities.
A management letter report showing only the findings with detailed descriptions of the results, assessments and recommendations. This report can be used for communication with the auditee.

Additional features not implemented yet are benchmarking functionalities in order to (1) compare different organizations or (2) compare the internal controls status for an organization at different times. (1) could be interesting to derive best practice system controls since a relative comparison between organizations can be achieved. For this purpose key performance indicators need to be identified in order to compare organizations within a homogeneous comparison group. These key performance indicators could be e.g. indicators reflecting a system's size like number of users or number of transactions. (2) could be interesting for high level management in order to assess the internal control system quality in variation of time on a CFO or other governing bodies level. A benchmarking requires a central audit result database which will be synchronized with local instances of the software.

5.6. Evaluation

The prototypical software contains around 30 control tests for the SAP R/3 system. We implemented the test of system related internal controls in the basis maintenance area and the purchase-to-pay process. The software prototype has been tested at a few systems. The data tables containing the relevant information to be audited have been downloaded to the ERP AuditLab through the Remote Function Call Mechanism (RFC) which turned out to be a very efficient method with only few manual activities.

Following main results have been revealed during testing:

(1) The automated controls testing within the framework as explained in this paper in general works without problems. Control tests could be added without implementing any additional source code.

(2) If an automated control test did not work appropriately it was not always easy to evaluate the error. Since the rules can have an arbitrary complexity, audit trails of the control test accomplishment should be improved.

(3) Some control tests showed a bad performance since data queries took a long time. We derived a need to integrate database indexes into the Data Object and Data Object Definition approach as described above.

(4) Our approach with Simple Rules and Rule Dependencies as explained above cannot cope very well with control tests needing to compare values with a comprehensive list of to-be-values. A mechanism with static tables with to-be-values needs to be implemented.

Our first evaluation of the prototypical software framework was rather technically oriented. In a next step we need to verify the prototype on a more functional level together with non-technical oriented persons or auditors.

6. Summary

This paper introduced a framework for testing internal controls on the application level. The framework allows testing controls with a highly automated approach. An important requirement was to allow implementing further control tests without extending the prototypical software on the source code level. New control tests can be easily “put” in the ERP AuditLab by defining rules on the customizing level. Next steps to improve the framework within the prototypical software will be synchronization features with a server allowing the central storage of audit results for benchmarking purposes and the download of newly implemented control tests. With this synchronization functionality, we plan to build up a central infrastructure to bundle deep technical system related audit knowledge for several different ERP systems. This should result in an accelerated and quality-improved audit process.

7. References


[23] H. Ye, Y. He, “A continuous auditing model based on Web services”, 7th WSEAS Int. Conf. on Applied Computer & Applied Computational Science (ACACOS ’08), April 6-8, 2008.