Towards Audit 2.0 – A Web 2.0 Community Platform for Auditors

Nick Gehrke
University of Hamburg
School of Business, Economics and Social Sciences
nick.gehrke@wiso.uni-hamburg.de

Patrick Wolf
Frankfurt School of
Finance & Management
p.wolf@frankfurt-school.de

Abstract

Generalized Audit Software (GAS) is a certain type of data analysis software that is increasingly used by auditors to improve the effectiveness and efficiency of audits. To date, the development and distribution of GAS follows traditional ways, that is the software is developed by a single company without the direct participation of the users, and the completed product is sold as closed-source with no rights for any modifications. We argue that this traditional model hinders the further adoption of GAS due to the artificial barriers imposed on the users (e.g., vendor lock-in, lack of flexibility). Thus, this paper proposes a new form of GAS that is built upon the principles of Web 2.0 and aims at offering auditors a more open, more flexible, and less costly alternative to traditional solutions. The main artifacts of this new form of GAS are conceptualized in the paper and possible business models are outlined.

1. Introduction

In recent years, auditors are facing a growingly digitized business environment where paper-based transactions are being replaced by electronic ones and where accounting-relevant data is increasingly exclusive generated, controlled and managed by complex, integrated Information Systems (IS), without or with only little human intervention [1] [15] [4]. In addition to this growing digitization of the firm, auditors have to cope with continuing smaller audit-budgets and stricter regulatory requirements that demand for more effective and efficient audits [2] [5].

Computer-Assisted Auditing Techniques (CAAT) are seen as an important means for auditors to respond to these fundamental changes and requirements. In its broadest sense, the term CAAT refers to any use of Information Technology (IT) within the audit process [5]. This paper adopts a more narrow understanding, according to which CAATs are tools and techniques used by the auditor to (semi)automate tests of controls, substantive tests and analytical procedures [2] [5] [6]. Depending on where, when, and how this testing is actually performed, different types of CAAT can be distinguished. The most frequently used type is Generalized Audit Software (GAS) [2] [5]. GAS offer the auditor various functions to extract accounting-relevant data from different sources (e.g., databases, files) and to perform complex analysis on this data. They operate almost independent from the client’s systems which is a main advantage over other CAATs.

To date, the development and distribution of GAS follows traditional ways, that is the software is developed by a single company without the direct participation of the users, and the completed product is sold as closed-source with no rights for any modifications. This traditional, proprietary model results in a strong dependency of the auditor from the software vendor, regarding ongoing updates, modifications, and support. Based on the findings of some recent empirical studies [7] [8] [9] [10], we argue that the shortcomings of the traditional model negatively influence effort expectancy and, thus, hinder the further acceptance and adoption of GAS by auditors.

The purpose of this paper is to present a new form of GAS that builds upon the principles of Web 2.0. This new form of GAS (GAS 2.0) aims at offering its users a more open, more flexible and less costly alternative to traditional solutions. In our new form of GAS, auditors are both consumers of audit-related services and producers of them (prosumers). They can design small audit routines, programs or services called auditlets for the purpose of testing (e.g. internal application controls) and share them with other auditors through a central server platform. In this way, audit knowledge, once established as auditlets, can be efficiently reused by other auditors without the need for reinventing the wheel over and over again.

The remaining paper is structured as follows: First, a brief overview of related work is given. Subsequently, the role of GAS in an Audit of Financial Statements is explained. In section 4, important deficiencies of traditional GAS are identified and potential benefits of GAS 2.0 are highlighted. In section 5, relevant requirements pertaining to GAS 2.0 are presented, derived from interviews with several
auditors and consultants. Section 6 presents the conceptualized artifacts of GAS 2.0. The paper ends with a description of possible business models and an outline of further areas of research.

2. Related Work

We reviewed the literature pertaining to computer-assisted auditing, Web 2.0 and knowledge management as GAS 2.0 draws concepts from all these areas.

In terms of computer-assisted auditing and Web 2.0, no comparable idea of sharing codified audit knowledge over a web-based auditor community could be identified. The idea seems rather new, considering the existing research topics, like full or partial automation of audits (e.g. [22], [23], [24]) or continuous auditing (e.g. [13], [25], [26]). The main difference of our approach to existing ones is the central idea of an auditor-community sharing codified audit-knowledge (auditlets) through a central server platform and the strong link to Web 2.0 principles (explained in section 4).

The area of knowledge management is relevant for GAS 2.0 as sharing of codified audit knowledge constitutes a key aspect of our idea. The question here is how a critical mass of users can be attracted to the auditor-community and how auditors can be motivated to participate actively in creating and sharing auditlets.

Partial answers to this question can be found in scientific papers which study the factors influencing the motivation of individuals for knowledge sharing. Several such studies have been published in recent years (e.g. [16], [17], [18], [19], [20], [21]). However, none of the identified papers specifically deals with knowledge sharing in the context of auditing. Thus, we used the papers to derive requirements pertaining to knowledge sharing in the context of GAS 2.0 (see section 4.2) and to identify ways of motivating auditors to participate. Further research needs to be carried out as of what specific combination of incentives works best in the case of GAS 2.0 (see section 8).

3. The role of GAS in an Audit of Financial Statements

The objective of an audit of financial statements performed by an external auditor is to express an opinion on the true and fair presentation of the financial statements of an entity and the conformance of these statements with an identified financial reporting framework (e.g. IFRS) [14].

The audit process consists of several phases (Client Acceptance, Audit Planning, Testing, Reporting and Closing), whereas the phase most intensively supported by GAS is Testing. This phase consists of performing the defined test procedures, gathering the resulting audit evidence, and deciding on whether additional test procedures are needed. With regard to their purpose, test procedures can be classified into tests of controls and substantive tests.

Tests of controls are performed to obtain evidence whether the internal controls are suitably designed to prevent or detect and correct material misstatements and whether the internal controls operated properly throughout the accounting period. Substantive tests are analytical tests (e.g. comparison of actual and to-be-values), tests of transactions (e.g. inspection of corresponding system records), and tests of balances (e.g. to identify misstatements in general ledger balances) [14].

GAS provide numerous functions to support both tests of controls and substantive tests and can be applied whenever electronic data needs to be analyzed to support human judgment or perform automated evaluations. Some examples of audit analysis supported by GAS are: (1) Auditing a configuration file regarding password security (e.g. length and complexity of passwords) and comparison to to-be-values; (2) Auditing vendor master data audit trail files regarding the manipulation of vendor bank accounts; (3) Auditing purchasing transactions and filtering on transactions where goods receipt took place before purchase order creation.

4. Deficiencies of traditional GAS and potential benefits of GAS 2.0

In this section, we describe the most important deficiencies of traditional GAS, like ACL (Audit Command Language) or IDEA (Interactive Data Extraction and Analysis), and how these deficiencies are addressed by our proposed new form of GAS that builds upon the principles of Web 2.0 ('GAS 2.0').

4.1. Deficiencies of traditional GAS

In order to understand the problems and difficulties when using traditional GAS, we performed open interviews with three (IT)-auditors of a big four audit company. All interviewees had at least two years of IT-audit experience. They all applied GAS software during past audits several times and, in this context, used the programming features of GAS. Regarding their answers, we found three main problem areas.

(1) Typically, clients provide the data required for GAS-supported auditing as Fixed-width files (e.g. SAP reports) due to the fact that these files can be
easily created using standard system functions without the need of direct interaction of the auditor with the client's systems. In order to extract the relevant data from a provided file, an appropriate filter needs to be created within the GAS that separates relevant from non-relevant data (e.g. formatting data). Depending on the file structure, creating such a filter can be a complex and time consuming task, requiring specific knowledge of the build-in scripting language of the respective GAS in use. As a consequence, auditors not having this knowledge may not be able to use traditional GAS in more complex but often occurring audit situations in an effective way.

(2) Once appropriate file filters have been created, traditional GAS do not support any efficient sharing of the created filters (or other audit routines) with other auditors. Usually, a "work-around" is required consisting of sending the project file or the extracted source-code of the audit routines by email to interested colleagues. However, this approach is limited to personal social contacts only and relies strongly on manual activities.

(3) Audit routines developed by auditors often contain parameters coded in the source code (e.g. the actual financial year to be audited). Thus, reusing the audit routines often requires adjustments in the source code which is inconvenient and prone to error. This hinders the spread of audit knowledge incorporated in existing audit routines.

The aforementioned deficiencies of traditional GAS above led us to the idea of applying the principles of Web 2.0 to GAS. The next section gives a short overview of the principles of Web 2.0 and describes the benefits of applying them to "GAS 2.0".

4.2. Potential benefits of Web 2.0-based GAS

The term Web 2.0 derives most of its popularity from a widely cited article published by Tim O’Reilly in 2005 [11]. This article highlighted the main principles underlying Web 2.0 applications (e.g. Wikipedia, Facebook, YouTube, Flickr) which are shown in Table 1. In 2006, O’Reilly published a subsequent article on the subject with the following compact definition of Web 2.0 [12]:

"Web 2.0 is the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to get better the more people use them [...]"

<table>
<thead>
<tr>
<th>Principle</th>
<th>Characterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web as a Platform</td>
<td>Web 2.0 applications build on the shared and open infrastructure of the Web. Their services are accessible from anywhere and across different devices.</td>
</tr>
<tr>
<td>The Long Tail</td>
<td>Web 2.0 applications aim at making niches more accessible to their users and exploiting the economic value of these niches.</td>
</tr>
<tr>
<td>Data is the Next Intel Inside</td>
<td>Web 2.0 applications seek to create a unique source of data that is hard to copy or recreate. The data is their chief competitive advantage.</td>
</tr>
<tr>
<td>Users Add Value</td>
<td>Web 2.0 applications shift the task of creating and updating content to their users. They harness the collective intelligence of their users to create a unique source of data.</td>
</tr>
<tr>
<td>Network Effects by Default</td>
<td>Web 2.0 applications generate network effects not only from the direct contributions of the users, but also from the side-effects of the use of the offered services.</td>
</tr>
<tr>
<td>Some Rights Reserved</td>
<td>Web 2.0 applications are build upon existing standards and use licenses with as few restrictions as possible. They aim at lowering the barriers for adoption and allow for &quot;hackability&quot; and &quot;remixability.&quot;</td>
</tr>
<tr>
<td>The Perpetual Beta</td>
<td>Web 2.0 applications are collections of constantly improved services that never leave the development stage. Releases are done early and often (monthly, weekly, or even daily) and the users are engaged as real-time testers.</td>
</tr>
<tr>
<td>Cooperate, Don’t Control</td>
<td>Web 2.0 applications are build upon &quot;lightweight programming models that allow for loosely coupled systems&quot; [11]. This allows for the creating or assembling of services in a quick and cost-effective way.</td>
</tr>
<tr>
<td>Software above the level of a single device</td>
<td>Web 2.0 applications are not limited to a single device. Their services are used across different platforms and devices via Web-Browser.</td>
</tr>
</tbody>
</table>

Table 1. Principles of Web 2.0

In Web 2.0-based GAS or "GAS 2.0", the community members (auditors) develop the actual content (auditlets) by themselves and the platform supports the exchange of the auditlets among the community members. The platform is separated from the actual audit knowledge and provides only the required infrastructure. By contrast, in the traditional "GAS 1.0" approach, platform and content are most often bundled in monolithic software. Here, the
potentials for auditors to develop own audit routines are limited and especially automated sharing of this knowledge is not supported: developed audit routines need to be distributed manually (dashed lines) which is often prone to errors (see figure 1 for illustration).

Figure 1: The ‘GAS 2.0’ approach (left) vs. the ‘GAS 1.0’ approach (right)

The main potential benefits of GAS 2.0 are:

1. If an auditor does not possess of the required technical knowledge to create a specific, complex audit routine in the GAS, he may use the platform to search for an existing auditlet representing the required audit routine. If such an auditlet exists, he may download and use it (probably against payment). If no such auditlet exists, the auditor may ask the community members for (payed) help via the platform (that is to create the required auditlet). Provided the right incentives exist and the community has reached a critical mass, the community members will usually react faster than traditional GAS vendors and, in addition, will fulfill more specific needs (long tail effect).

2. Due to the fact that sharing of audit-specific knowledge (auditlets) is at the heart of GAS 2.0, it is expected that applications based on this idea will allow for a more efficient sharing of such codified knowledge among a broader audience, allowing for significant gains in work efficacy.

3. The idea of auditlets aims at clearly separating variable parameters (e.g. fiscal year) from more static source code to allow for easier and more extensive reusability.

Table 2 applies the Web 2.0 principles to GAS 2.0:

<table>
<thead>
<tr>
<th>Principle</th>
<th>Characterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web as a Platform</td>
<td>Central to GAS 2.0 is a Web-based platform that supports the exchange of auditlets among interested parties (e.g. auditors, software developers, freelancers).</td>
</tr>
<tr>
<td>The Long Tail</td>
<td>According to our idea, auditlets possess of structured interfaces and can be created and shared by interested parties. If an auditor needs a specific auditlet for a nonrecurring event, he can make his need public to the community via the platform. Subsequently, community members may pick up the need and provide the auditor with the required auditlet. By contrast, traditional GAS is often distributed as closed-source. Thus, only the GAS vendor is able to perform functional improvements, usually based on strategic and financial considerations. Functions for which most customers do not want to pay will therefore not become part of the standard version of the GAS.</td>
</tr>
<tr>
<td>Data is the Next Intel Inside</td>
<td>The unique source of data of GAS 2.0 are the auditlets created by the community and the data associated with these auditlets (e.g. auditor comments, evaluations).</td>
</tr>
<tr>
<td>Users Add Value</td>
<td>As described earlier, applications classified as GAS 2.0 shift the task of creating and updating audit-specific content to their users. They harness their collective intelligence and motivate them to participate through appropriate incentives (e.g. community recognition, monetary compensation).</td>
</tr>
<tr>
<td>Network Effects by Default</td>
<td>GAS 2.0 generate different side-effects from the use of the offered services, e.g. &quot;Most frequent downloaded auditlets&quot;, &quot;Best evaluated auditlets&quot;, &quot;New auditlets in the current week&quot;, &quot;Most active users&quot;, &quot;Best rated users&quot;, etc.</td>
</tr>
<tr>
<td>Some Rights Reserved</td>
<td>In order to allow for the exchange of auditlets, structured interfaces are defined for different types of auditlets (e.g. auditlets for data acquisition, see section 6). Based on these interfaces, auditlets may be combined to more complex auditlets.</td>
</tr>
<tr>
<td>The Perpetual Beta</td>
<td>Our idea envisions that releases of auditlets are done early and often. Via the platform, users can evaluate the offered auditlets (e.g. with regard to security, performance, or integrity), notify the creators of existing errors, or make suggestions for amendments.</td>
</tr>
<tr>
<td>Cooperate, Don’t Control</td>
<td>Due to defined interfaces, Auditlets may be combined / assembled as required in order to perform complex audit procedures. The search and download of auditlets is supported by the underlying platform and is carried out, in most parts, automatically.</td>
</tr>
<tr>
<td>Software above the level of a single device</td>
<td>Due to the use of a Web-based platform, GAS 2.0 is not limited to a single device. Auditlets can be exchanged across different platforms and devices via Web-Browser. They can be automatically downloaded and plugged in the auditor's client software to actually perform the audit.</td>
</tr>
</tbody>
</table>
5. Requirements for GAS 2.0

After having discussed the deficiencies of traditional GAS and the potential benefits of GAS 2.0, this section derives important requirements that the envisioned new form of GAS should fulfill. The presented requirements were partially derived from own thoughts and partially from interviews conducted with employees of an auditing company and an audit related consulting company. Before each interview, we presented the idea of the Web 2.0 auditor community shortly and explained the main properties (e.g. how small audit routines could be designed and shared using the community platform). We then asked each interviewee whether he/she would personally use such a community, whether he/she sees any hurdles and whether he/she has any specific requirements that should be fulfilled by the community.

5.1. Requirements related to the identified deficiencies of traditional GAS

Related to the identified deficiencies of traditional GAS (see section 4.1), we derived the following requirements for GAS 2.0:

<table>
<thead>
<tr>
<th>Deficiency</th>
<th>Derived Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need of specialists familiar with the build-in scripting language of the GAS in case of complex audit situations</td>
<td>Source code should not be used to design audit routines. Instead, editors like query generators should be used extensively. It is assumed here that such generators are more user-friendly and allow (in most cases) for a graphical design of complex audit-related queries.</td>
</tr>
<tr>
<td>Inefficient sharing of existing and proved audit routines among users</td>
<td>Here, we derive the need for a system supported sharing of already developed audit routines, which can be further broken down to the need for community mechanisms based on Web 2.0 principles.</td>
</tr>
<tr>
<td>Reusing audit routines often requires adjustments in the source code which is inconvenient and prone to error</td>
<td>With regard to this deficiency, we derive the requirement for a strict framework for audit routines or – how we call it – auditlets. The framework needs to define structured interfaces to segregate the implementation for data acquisition, data access, data aggregation, data analysis, auditing against parameters (to-be-values), and reporting of audit findings.</td>
</tr>
</tbody>
</table>

Table 3. Requirements derived from the identified deficiencies of traditional GAS

5.2. Requirements related to motivating auditors to engage in the auditor-community

One critical success factor for GAS 2.0 is to build and retain an active community of auditors (and other interested parties) who regularly create and share auditlets and engage in additional community activities (e.g. evaluation of offered auditlets).

According to [17], the primary trigger for engaging in knowledge transfer activities (like the ones mentioned above) is intrinsic and extrinsic motivation. Intrinsic motivation "[…] refers to engaging in an activity for its own sake, out of interest, or for the pleasure and satisfaction derived from the experience" [17]. Intrinsic motivators may be the joy of helping others (altruism) or knowledge self-efficacy.

Extrinsic motivation, on the other hand, is based on the perception of the value associated with knowledge exchange. From a socio-economic perspective, auditors and other parties may engage in creating and sharing auditlets if the perceived benefits of participation equal or exceed the resulting costs (e.g. time taken, mental effort, etc.) [17].

Here, we derived the following requirements

(1) In order to motivate auditors and other parties to engage in creating and sharing auditlets, GAS 2.0 should incorporate appropriate incentive schemes that address intrinsic as well as extrinsic motivation. For example, software developers may be motivated extrinsically by the possibility to collect a fee each time an offered auditlet is downloaded or used (via micropayments, see also section 7.2). Auditors may be motivated extrinsically by offering them with the possibility to gain a strong reputation within the auditor community. Another motivating factor may be the ease of use of the platform regarding the search and retrieval of auditlets [18].

(2) For attracting auditors and other parties to the auditor community, a critical mass of activity is required: "without critical mass, the perception of the usefulness of the knowledge-sharing system will inhibit its use" [16]. Thus, we derived the requirement that GAS 2.0 should incorporate mechanisms to prominently feature interesting activities within the community and to promote newly uploaded auditlets. These mechanisms ensure that the platform comes across as highly dynamic and contribute to the positive appeal of the platform to new users.
5.3 Requirements related to the reliability and privacy of auditlets

With regard to the reliability and privacy of auditlets, we derived the following requirements:

(1) Since the auditor community is open for any user, offered auditlets are basically not trustworthy, might be defective or not fully reliable. Thus, auditlet provider should have the possibility to let their auditlets electronically signed by a trusted independent third party. The role of this party is to certify the auditlet based on a thorough examination of the auditlet (e.g. with regard to any contained malicious code). The certificate aims at strengthening the trust of the users in the offered auditlets and might allow auditlet providers to differentiate themselves from others.

(2) Our interviewees often argued that they would not want to share all developed auditlets with a public community since some auditlets incorporate exclusive audit knowledge which constitutes a competitive advantage for them. We therefore derived the requirement that closed user groups (corporate communities) should be possible which are isolated from the public community and allow for sharing only among authorized members of these closed user groups.

(3) In general, our interviewees were the opinion that a public Web 2.0-based auditor community could produce interesting and useful auditlets. However, with regard to the transfer of a publicly available auditlet to a closed user group (corporate community), they recommended a prior investigation and approval process in order to ensure that only auditlets of a certain (approved) quality become available in the corporate community. Thus, auditlets marked for transfer from the public into the private realm, should be moved to a quarantine area within the corporate community first, until the final approval is given.

6. Concepts of GAS 2.0

This section presents and discusses the main concepts of GAS 2.0: a precise definition of the term "auditlet" is given and the architecture is described.

6.1. Definition of Auditlets

An auditlet is a tiny and isolated audit procedure incorporated in a software component which processes a task important for an audit related test or analysis. Auditlets can be automatically plugged in the auditor’s client software platform and are transferred via a central server platform. In order to allow for an automatic plug-in, auditlets must adhere to defined interface structures. Auditlets can depend on each other. These dependencies are managed by the central server platform and the auditor’s client platform.

The term auditlet is analogue to the well known term applet which means a software component that runs in the context of another program.

6.2. Architecture Overview

Figure 1 shows the general components and principles involved in our GAS 2.0 community. Depending on their respective skills, auditors (and other parties) can be both auditlet developers and auditlet users. They can upload and download auditlets from a central server platform, serving as an auditlet storage or catalogue. Auditlets are continuously developed and extended by the community members.

The enhancement of audit functionality is shifted from a software vendor to a community, which basically adheres to the principle of crowdsourcing, denoting the outsourcing of a creative development process to an amorphous crowd of specialists and not to a (set of) predefined business partner(s) in order to harness the collective intelligence of the members of the crowd. Two example auditlets are shown in figure 2: While the first auditlet is used for auditing the SAP R/3 report RSPARAM containing basic technical settings for the system, the second auditlet is used for auditing the report RFKABL00 that contains the vendor master data change log.

![Image of Auditors as Prosumers in the Community](image.png)
5.3. Technical Architecture

In the following, we show the technical design of the client platform. It is explained how automated testing using the new form of GAS is achieved and what software components are involved. We also clarify what types of auditlets are possible and how auditlets depend on each other. Please note that we will not address all requirements discussed above (corporate community, digital signature, approval processes etc.) since this section aims to show only the basic plug-in oriented auditlet principles.

![Diagram of the audit process](image)

Figure 2. Technical architecture of the audit process

Figure 2 shows how testing is performed, starting with the acquisition of the data and ending with the interpretation of the findings. We designed the system to have six important components. Please note that five of the six components are tagged with the hint “auditlet” in the figure meaning that these components are specific types of auditlets. If one of these components is parameterized by an auditor, the parameterized component can be considered to be an auditlet. Please further note that an auditor usually does not implement the auditlets with source code of any programming language. Instead, each type of component can be parameterized within defined functionality. The auditlets to be shared among the community can be understood as parameterizations of one of the components. Thereby, an auditlet needs to define dependencies to other auditlets since a single individual auditlet only processes a specific task within an audit of a file or data set. The components or parameterized auditlets can be described as follows:

1. **Report / Data Filter.** These auditlets only transform data to be audited into a table oriented relational format. The data to be audited exists in an IT-system and needs to be extracted for audit purposes. Although a direct database connection could just extract data already being in a relational format, auditors often have only ‘normal’ access to systems. This issue has been reported to us during interviews with a big-four company. Often, auditors extract standard text reports which are intended to be read by humans and not machines. These reports need to be parsed and rearranged to a relational structure. Even if a direct database connection is possible, understanding relational database tables is much more difficult than understanding a system generated report that seeks to be self-explanatory. The parameterizations of this component / auditlet are e.g.: Position of important data fields in the report; Hierarchies of displayed data in the report.

2. **Data Access Object.** This component is needed to provide a relational access to parsed data. It is only a technical issue. Auditlets cannot be defined for this component.

3. **Data Mapper.** If a report has been parsed, the data fields in the report might depend on a specific proprietary IT-system. A data mapper can be used to translate data fields to other data fields in a common format. Data mappers are ‘translation services’ for operator auditlets (see below). The parameterizations of this component / auditlet are only lists mapping data fields from the proprietary report format to any other format.

4. **Operator.** The operator aggregates or filters data as needed for auditing. He depends on data fields of the data mapper. Please note that operators could also operate directly on the data fields of a parsed proprietary report without using a data mapper. In this case, the operator is also proprietary. Data mappers are used if a generalization of the operator beyond a specific IT-system is intended. It depends on the auditor, if operator auditlets are designed with or without data mappers. Please note that operator auditlets could depend on more than only one data mapper or table. In this case a multiple dependency exists and multiple data sources need to be prepared by upstream auditlets. The parameterizations of this component / auditlet are an arbitrary set of SQL-Statements and a defined hierarchy of these SQL-Statements.

5. **Test-Parameterizer.** Data aggregated by operator auditlets need to be compared to to-be-values or to an expected audit result. The Test-Parameterizer contains rules and to-be parameters for evaluating the aggregated data of operator auditlets. The parameterizations of this component / auditlet are rules for evaluating aggregated data of operator auditlets and to-be-values.
6. Result Interpreter. This component contains textual formulations to document and communicate the audit results in a natural language form. Audit results, findings and recommendations are pre-formulated and applied depending on the rules of an upstream test-parameterizer auditlet. Within these formulations, placeholders can be used to display any results found in the data. The formulations can be used for reporting reasons. The parameterizations of this component / auditlet are a set of predefined formulations and a mapping to the rules of an upstream parameterizer auditlet.

Please note that auditing data or reports of a system does not mean that all types of auditlets must exist in order to accomplish the full cycle. It might be that there is only a report filter auditlet which by itself is deemed valuable by the auditor community. It might also be that there are alternative result interpreter auditlets for one test-parameterizer auditlet. An auditor might not agree with the result interpretation and decide to use the alternative result interpreter auditlet instead. Auditlets can be combined and alternatives can be developed on each level for the same data to be audited. The community of auditors shall choose and use the best auditlets and functionality shall evolve through time.

6.4. An Example

Let us consider a simple example. In advanced ERP systems like SAP R/3 specific financial accounts can be customized for automatic postings only. If an account is customized like this, manual general ledger postings are rejected by the system. Instead, only automatic postings triggered by specific transactions like goods receipt or invoice processing can accomplish a posting. Some accounts like goods receivable / invoice receivable (GR/IR) or accounts payable (AP) are usually automatic. An interesting issue for an auditor is to audit the change log of account master data to evaluate if accounts have been switched from automatic to manual and back. This could give hints to unapproved manual postings or business process problems.

![Figure 4: Change Log for G/L accounts in SAP R/3](image-url)

The figure above shows the account master data change log. As one can see, the same user switched off the automatic posting flag and switched it on again within 10 minutes. In the following the technical audit process with appropriate auditlets is shown:

1. The report filter auditlet knows the structure of the report and parses it. The auditlet is parameterized to ‘cut the relevant fields out of the report’. It knows the relevant positions of the data fields and how to filter text rows.

2. The data access object encapsulates the data set as a result of the report filter auditlet. It contains e.g. the fields Time, Date, Account, User, Fieldname, NewValue, OldValue within a table structure.

3. A data mapper auditlet might exist or not. In this case we make no use of it.

4. The operator auditlet filters and aggregates the data. It contains e.g. SQL statements which (1) select entries only with FieldName=’Automatic’ and (2) selects only entries where the ‘Automatic’ flag has been turned off and on again within T days.

5. The test-parameterizer auditlet defines T to be 5 days and two possible outcomes: (1) the switch on/off change has been accomplished by two different users or (2) has been done by the same user.

6. The result interpreter auditlet interprets the outcome (1) as weak finding and the outcome (2) as severe finding and assigns a predefined text A resp. B to the two outcomes.

7. Possible Business Models

The proposed Web 2.0 auditor community consists of highly specialized professionals. The question arises which business models could be implemented to maintain the software platform in order to allow auditors to develop auditlets. We propose three different business models which could be combined to an aggregated model.

7.1. Corporate Community

As already explained in the requirements section, audit firms want to keep their own developed auditlets within their company. They need to set up an own
corporate community. A business model could harness this need and act as follows: The client platform for the auditors to develop and use auditlets would be free for download and use by the public. The central server platform distributing and indexing the auditlets would be free for use as well. However, it would not be available for free download and installation in a separated environment. If a company wants to set up an isolated corporate community, it needs to pay for a private instance of the server platform. Pricing models could be coupled to the intensity of use which would reduce the risk that an expensive platform is bought but not used.

7.2. Micropayment Approach

The micropayment business model approach aims to directly remunerate auditors or other parties who develop and share auditlets. Developing auditors ‘own’ their auditlets and other auditors who want to use them need to pay for it. A portion of the payment will be directed to the operator of the central server platform. This approach is problematical since auditors might not want to pay for auditlets they do not know. Thus, digital signatures and certificates for auditlets become even more important in this scenario.

7.3. Continuous Auditing Platform

Continuous Auditing is "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" [13]. Continuous auditing means scanning systems and data on a frequent basis according to corporate compliance rules [3]. It is usually performed from within a company, typically by the Internal Audit department. Auditlets could be used to support continuous auditing. The main difference here by comparison to the normal auditor client platform is that the auditlets need to be triggered automatically and not, as in traditional audit engagements, ad-hoc. The business model aims to charge a company if it wants to use the client software platform in the 'continuous auditing mode'.

8. Summary and Outlook

This paper introduced and designed a new form of GAS harnessing the power of a Web 2.0 auditor community. Auditors can develop and share small audit procedures called auditlets and thus enrich the functionality of the software according to a collective intelligence approach through the community.

First, we discussed the deficiencies of traditional GAS and potential benefits of Web 2.0-based GAS. Subsequently, we derived requirements from interviews with several employees of different audit and consulting companies. Based on these interviews, we designed the technical architecture of our community oriented software and developed ideas for appropriate business models.

In the next research step, an evaluation of the design of our platform is needed. We intend to implement a prototypical platform for the development and exchange of auditlets that will be subject to further scientific evaluation.

Additionally, further research activities need to address a major concern with regard to establishing the auditor community: According to [19], "the successful implementation of a KMS [Knowledge Management System, like GAS 2.0] requires the active participation by a "critical mass" of users almost from the start of the implementation" [19]. The more users the GAS 2.0 platform has the more benefits it offers to auditors and other parties and the more attractive it is for them to join and participate (so-called "network effect"). Here, the question arises as how the start-up problem can be overcome, referring "to the costs and practical difficulties of attaining a viable user community starting from zero" [19]. Some recommendations can be found in the literature (e.g. to offer a well designed user interface and positive incentives or rewards to early users, to focus on a smaller subset of users to begin with [19]). However, it needs to be studied in more detail what the most effective combination of these and other measures (e.g. to offer a certain set of auditlets right from the beginning) in the context of GAS 2.0 is. The main objective here is to reduce the risk of a complete failure of the platform due to the inability to attract a critical mass of actively participating auditors and other parties.

9. References


