Agile Audit

Time constraints are an integral part of every auditor's work. Audits must finish on time. Using the allotted time efficiently is a major concern. Agile audit is primarily about increasing the efficiency mainly of complex audits by parallelizing tasks, eliminating or mitigating bottlenecks, and assigning time to various tasks that is proportional to each task's importance.

The term "Agile" usually refers to software development and emphasizes individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.[1]

"In Agile models, design and specification documentation are kept to the bare minimum required, and the major part of documentation is created at the operations and support levels."

Its appropriateness to complex systems is also stressed in the Certified Information Systems Auditor® (CISA®) reference manual: "The term 'agile development' refers to a family of similar development processes that espouse a nontraditional way of developing complex systems."[2] As an example, a project to develop personalized web-based services that had been going on for three years had an extremely negative review a month before the deadline, since nothing had been developed. A new team of four people, including a new leader, was asked to take over and they were able to meet the deadline with impressive results. The new team focused on immediately developing a working system without bothering with long meetings, documentation and formal reviews. They conducted thorough internal trials to identify and immediately correct any issues and created documentation after the system was working and stable.

Audit, on the other hand, has traditionally used fairly strict standards and frameworks, resulting in rather rigid audit engagement constraints that, essentially, represented projects. IT projects have similarly inflexible models. However, they have evolved from the formal waterfall model, which has strict steps, to less formal, but very often more efficient, models. These more efficient models are usually collectively known as Agile. In the rigid models, proportionally much more effort is put into design and specification documentation. In Agile models, design and specification documentation are kept to the bare minimum required, and the major part of documentation is created at the operations and support levels, e.g., user manuals, which occur much later in the system life cycle.

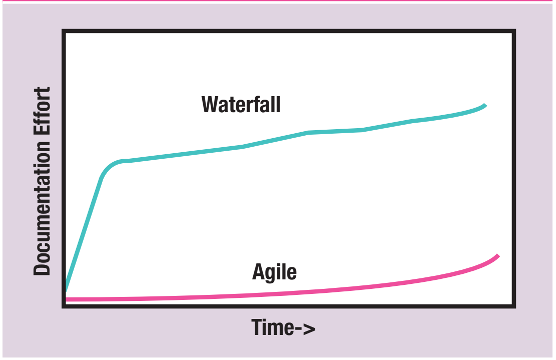
The documentation effort for the waterfall and Agile methods is illustrated in **figure 1**. The steep slope in the beginning of the project for the waterfall method is due to project overhead, such as project planning as well as specifications (both high-level and detailed) and design. After the design is completed, relatively little documentation is required until near the end,

Do you have something to say about this article?
Visit the *Journal* pages of the ISACA® website *(www.isaca.org/journal)*, find the article and click on the Comments link to share your thoughts.

Spiros Alexiou, Ph.D., CISA
Is an IT auditor who has been with a large company for nine years. He has more than 20 years of experience in IT systems and has participated and led both projects and audits employing Agile methods. He can be reached at spiralexiou@gmail.com.

where support documents need to be produced, such as user manuals.

Figure 1—Documentation Effort as a Function of Project Time for the Waterfall and Agile Models



Source: S. Alexiou. Reprinted with permission.

In contrast, for Agile, the documentation requirements are much lower than the waterfall documentation requirements—practically zero until near the end of the project where, again, support documents such as user manuals need to be produced. Agile is much more efficient in that during the final state, documentation is thoroughly and formally captured, not at the initial or intermediate stages of the final deliverable. Documentation is only created when it is needed and, ideally, in the form that is needed, such as comments in the actual code.

In Agile models, for software development and other endeavors, the project team has more freedom and initiative to make adjustments as the project progresses. This is especially pertinent to audit, since the goal of the audit is not to serve its own methodology, but to add business value. This, in turn, if properly run, can result in higher efficiency and better results. For example, in a conventional (waterfall) project to, say, create a new IT system, one would typically first deal with specifications, then design, then development/implementation. Testing and acceptance would be last. An Agile approach would be quite different—a distinction that can be seen even in a very different project, such as an audit. In an Agile audit, one would focus on identifying and rapidly beginning testing the issues that carry the most risk, just like Agile software development would focus on creating a working prototype that would be subsequently improved. This principle is often referred to as Thompson's rule for first-time telescope makers, which states that "It is faster to make a four-inch mirror then a six-inch mirror than a six-inch mirror."[3]

The term "Agile audit" has been used before this article, and with more or less different meanings.[4, 5, 6, 7, 8] It is necessary to briefly review these meanings to distinguish them from the meaning "Agile audit" is given in this article.

"In an Agile audit, one would focus on identifying and rapidly beginning testing the issues that carry the most risk."

In one case, the following realization is expressed: "And yet, our audit cycle times can be longer than desired. Our output may be different from what our stakeholders expected. Our quality assurance processes may introduce constraints to efficiency that fail to produce more value-added insights." This led to a search to "improve internal audit's adaptability and response time" and "a way to standardize our approach to oversight of strategic projects and gain our stakeholders' acceptance of our role."[9] While the starting point is the same, the concept of agility, as used here, is very different. In the second case, agility is also used in a different context, that of keeping better track of the business impact.[10]

The next example is much closer to the definition used here, but also involves other concepts such as self-assessments and partnership with management that may also be applied to non-Agile audits.[11] Another example shares many of the concerns of this article, but stops short of exactly defining the main distinctive features of an Agile audit.[12] The last example recognizes the four audit phases (planning, fieldwork, response and final) and the temporal and logical separation they produce (they are called "Toll Gates" in that paper), and strives to be agile within these constraints, whereas this article calls for blurring or eliminating these formal Toll Gates.[13]

In this article, Agile audit is given a very concrete meaning that is distinct from the previous references. Specifically, it refers to blurring or altogether abolishing the sacrosanct temporal separation between planning and fieldwork. This means the end of planning is not necessary for fieldwork to start or for data to be requested, and tasks may be run in parallel. In addition, the production of a formal planning document is no longer required. This is motivated by the same considerations on documentation (which is what planning essentially is) as the previous remarks on Agile projects and their management. For instance, documentation may consist of an email to the auditee requesting specific information, plus the processing of that information and results of the test run, which are normally done at the fieldwork phase. Instead of documenting what will be requested and how it will be used, Agile audit documents what was requested and how it was used. Similarly, findings may be shared with the auditees before the final report,[14] but this is not a concern in the present article because this does not normally represent a major bottleneck.

What Is Wrong With Non-Agile Audits?

Even though IS auditors, audit projects and systems come in contact with Agile models, their own rules are often outdated. For instance, many audit departments specify that absolutely no data will be requested until the audit program has been finalized. Although this is well intentioned—to request only relevant data and to limit the perturbation of auditees who need to furnish the data—it is often counterproductive.

“Many audit departments have strict rules that no data are to be requested until the audit program is finalized.”

Typical audit engagements start with an exploratory phase in which the auditors familiarize themselves with the object of the audit. For instance, if an IS system will be audited, the auditor needs to understand what data it holds or processes, what the interfaces are, who uses the system, who the administrator is, and so on. These determinations need to be made before drafting an audit program. However, a few lines of data provide much more information and are a lot faster to comprehend than trying to read an entire manual of often very poor documentation or relying on the explanations of auditees who often have a very different focus from the auditor. In addition, most people, auditors included, learn better from examples than from dry documentation that needs to cover extremely rare cases on an equal footing with more normal cases (such cases may be encountered during testing, but in Agile audit, they are resolved then and there). Yet many audit departments have strict rules that no data are to be requested until the audit program is finalized. This has the following adverse effects:

- The audit program is drafted—for instance, to deal with data—by people who have never seen a single line of the data.

- As a result, the auditors must rely on their own interpretation of what they were told by the auditees, whose view of the system and the data is completely different from the auditors' perspective. This, in turn, means that aspects that are possibly important for the audit are left out of the briefing entirely because the auditees did not consider them interesting or relevant and the auditors did not know to ask about them.
- Even if the data, system, processes, people and functions are well understood, this still does not mean a rigid, written-in-stone audit program will result. Based on what the audit finds, there may be indications that more work is needed to cover risk that was initially unknown or underestimated. For instance, during the course of the audit, it may become clear that there are missing controls resulting in a high fraud risk.

“During an audit, the auditor is unaware of the priorities of the ultimate findings. The main goal is to discover and evaluate risk and propose controls for these areas of risk.”

- Getting all information before starting any fieldwork, even if one had enough information to carry out some steps from the very first day, creates a temporal bottleneck. In addition, when data are finally requested, getting the data involves further delays because the auditees who must provide the data may have other, higher-priority tasks or simply because extracting the necessary data as requested by the auditors may take time.
- Because planning involves limited information, risk may be over- or underestimated or missed altogether. Especially for complex audits, it is typically much easier to assess risk with detailed information than with only sketchy information. For instance, examining the data, its structure, and noting trends and exceptions can provide useful clues.

Once the audit program is finalized, often with misinterpreted information, precisely because no data were ever seen by the auditors, two things can happen. At worst, perhaps because of the poor understanding of the system and the associated risk, not to mention the approaching report deadlines, this will be a drive-by audit[15, 16] in which tickboxes will be checked and everything will be declared fine without looking deeply within the data or processes. At best, the auditors will realize their misunderstanding and will have to make a choice of revising the audit program and/or steps (and have to explain why the audit program was inadequate in the first place) or to note that some issues were not audited due to time or other constraints. Additional time would be lost if the auditors had done other work based on assumptions about the data they had never seen, such as writing software to analyze the data while waiting for the data. Audit inefficiencies are often a strong factor resulting in drive-by audits. Because time frames and deadlines must be respected, if time is spent inefficiently, it means that auditors will be tempted to perform the trivial tests that are sure to be completed on time rather than the more involved or complex tests dealing with many important risk factors.

During an audit, the auditor is unaware of the priorities of the ultimate findings. The main goal is to discover and evaluate risk and propose controls for these areas of risk. Audit programs often essentially assume the outcome is already known and try to specify not only the risk areas, but also how each step is to be carried out. As a result, audit programs are quite suitable for compliance or drive-by audits. Operational people tend to dislike such audits as they very rarely tell them anything useful. These types of audits tend to take up a lot of time and usually result in proposals that will mean more hassles with hardly any real value. That said, there are valid reasons for having some compliance audits and standard audit programs and models that are well suited to accommodate compliance audits. It is the application to all audits that may be outdated.

What Does Agile Audit Do Differently?

Instead of insisting on compliance with methodology and protocol, an Agile audit gives auditors much more freedom during the engagement phase to come in contact with the system, settings, data, and the people and processes being audited. This enables a much better understanding of the issues and risk to be addressed as well as how to go about testing them in detail (e.g., what tests to devise).

An Agile audit places much less emphasis on finalizing and documenting a formal audit program. In operational audits, an audit program is designed to address risk and some of these risk factors may crystalize during and not before the audit. For instance, it is very difficult, and wasteful, to devise all possible tests to look for fraud before having seen the data. In addition, risk that may have been identified as major may turn out to be minor or nonexistent due to strong mitigating controls. Also, risk that was considered much lower or not considered at all may be promoted as the biggest risk factors. Alternatively, as fieldwork progresses, it may be realized that risk that was not even considered before, either because the auditors did not know the area enough to ask or the auditees did not bring up the risk in the discussion, is quite important. As a result, an Agile audit program that focuses on tests and adapts to the work done and evidence uncovered may be much more suitable to addressing risk.

Agile audits, thus, address major bottlenecks in many audits. Necessary data, such as lists of system users from the system itself and an authorization database or file, can be requested and prepared by the auditees while the auditors are still trying to finalize remaining audit program steps. In addition, auditors can analyze data already collected while waiting for the audit team to schedule planning phase meetings with other auditees or the team members. By no longer insisting on strict temporal separation between planning and fieldwork, audit becomes more efficient. Tasks run in parallel (i.e., planning may be going on as the auditees collect requested data, or fieldwork is occurring while meetings to address remaining planning issues are being scheduled). For instance, if the audit team has already established that it will try and reconcile the user list generated by the system with the list of authorizations, does the auditor really need to wait to finish all other steps of the audit program? Waiting may mean finding time slots for more exploratory meetings with relevant, but possibly very busy, personnel as well as with audit team members in order to finalize the remaining steps before requesting the relevant data or actually running the reconciliation. This is illustrated by means of examples.

Real-world Examples

An audit involved checking billing records produced by IS devices. These records are used to charge for connection and volume and, as a result, they can be cross-checked by information collected at the network where probes are installed. Because of their complexity, probe records are not used for billing. This is a highly complex task involving sophisticated correlations, such as those among other complications such as traffic using different network segments (i.e., not being picked up by the same probe).

“An Agile audit needs and makes full use of the qualifications and expertise of each team member.”

As it turned out, there was only a single, very highly skilled auditor capable of carrying out this task. The auditor had a lot of experience with these systems before joining the audit team and undertook the task to cross-check billing records. Although initially it was required to specify detailed steps in the audit program, it soon became apparent that no one else could follow the steps. This made no sense. So, instead, the auditor created software to perform the cross-check and documented the functionality in his high-level code so that everyone could verify the findings. In addition, the auditor requested a small sample of data simultaneously with the audit announcement release. This enabled the auditor to start working on the cross-checking code immediately.

The result was that a highly complex audit was carried out in a very short time and with important results. As a spin off, the legacy of this audit was a full-blown system that could be used for monitoring billing on a permanent basis. For comparison, the same project was also assigned to an external company that took three times as long and concluded that the data were not good enough for their code.

A similar case involved a penetration test on company systems by an IS auditor. It turned out that getting the relevant permissions to conduct the test took a long time, as did preparing the penetration tools. The only reason the audit ran on schedule was because applying for the permissions and tool preparation started before and not after finalizing the audit program.

In yet another audit dealing with the security of system interfaces, each interface had its own security issues that could be determined only after detailed information was collected. Once again, the audit ran on time, but only because there was no strict temporal separation between planning and fieldwork. Specifically, knowing that audit team members were busy with other audits and finding a time slot to schedule a team meeting was not easy, the lead auditor prepared a document for the audit team with the list of interfaces and their function and high-level issues and simultaneously asked the auditees for relevant data. Some of these data came in while details of the interface architecture were being discussed and tasks were being assigned within the audit team. Thus, at the planning phase, the audit team had concrete information upon which to build.

Since the team had almost all the necessary data early, there were no bottlenecks associated with waiting for the data; in fact, because data were available, some tests finished early. It turned out that only one more set of data was needed to verify and assess the importance of a finding. This took substantially more time because auditees were busy with other priorities, whereas they had much more time when the initial data were requested. The audit team also adopted the practice of documenting and verifying each finding with the auditees as it came in without waiting for all findings to be completed. This was well received by the auditees, who could find a short window of time to discuss a single finding and had a much harder time finding the time to discuss a number of findings.

Misunderstandings, Risk and Pitfalls

Just as in Agile software development, an Agile audit is no substitute for risk identification and rational planning. Someone, usually the project manager or lead auditor, must come up with a rough skeleton of the audit program, which may be enhanced by the team. However, this is a very crucial first step in that a framework for discussion is established as fast as possible and the team has a concrete foundation on which to build. This is no different from standard audit programs, except that the steps need not be as detailed and definitely do not need to be formally documented. As noted earlier, audit programs tend to be rigid and written in a way that an auditor with minimal qualifications can follow. In contrast, an Agile audit needs and makes full use of the qualifications and expertise of each team member.

Agile audit does not do away with the need to document what was done. The difference is when the documentation is created and what it covers. Agile audit does not formally document in detail what it sets out to do and how it will go about doing it, but rather what it did and how.

A useful counterexample might be an audit conducted of a new business offering. In this case, planning involved questions on topics such as:

- Market issues
- Legal/regulatory environment
- Processes and systems
- IT support
- Business continuity issues

In this counterexample, typically a questionnaire would need to be constructed and filled out by auditees. The audit program is, essentially, this questionnaire. Hence, planning for what to ask and what evidence will be required to verify the responses is crucial and will be, by and large, also the final deliverable. Since this step will be present both in traditional and Agile audits, i.e., the work put into the audit program will be directly used in the results, Agile methods offer little or no advantages.

Identifying and prioritizing risk areas are key components in the audit program. It may be that, in a particular audit, little advantage is to be gained by using Agile methods. Agile audit is not an all-or-nothing method that the audit function must either always employ or always avoid. It is up to the audit team and the team leader to decide if task parallelization brings added efficiency or otherwise benefits the audit. For instance, if all necessary information is readily available and any requested meetings are immediately granted, then the importance of an Agile audit decreases.

Agile audits do not eliminate planning. An Agile audit substitutes a rigid plan with an adaptively improving plan that runs parallel to some fieldwork. Nor does an Agile audit eliminate quality control and documenting the work done and, especially, the findings. Null results must still be documented, as they provide assurance.

“Agile audit is not an all-or-nothing method that the audit function must either always employ or always avoid.”

Similarly, an Agile audit does not eliminate or diminish the importance of leadership. If anything, just like Agile software development, it places even more emphasis on leadership and team competence, as the auditors are not just executing strictly defined audit steps. They also design, modify and improvise these steps.

Last, it may be argued that a well-planned audit minimizes the interference of audit with everyday auditee work, while an Agile audit, which is

Figure 2—Pros and Cons of Agile and Non-Agile Audits

Audit Aspect	Agile	Non-Agile
Audit duration	Fast	Slow
Audit quality	Generally better, as more time can be devoted to material issues	More challenging, especially for complex audits
Audit complexity	Generally needs more highly qualified auditors	Can be executed by less qualified auditors who get a list of detailed steps
Audit flexibility	Easier to adapt to changes in risk evaluation	Needs a formal audit program revision
Leadership	Generally more important, at least during the initial stages	More democratic, as all team members participate more or less equally (in principle) to planning
Interference with auditees' time	In principle, may involve more short meetings with auditees	In principle, fewer, but longer meetings assuming all goes well

Source: S. Alexiou. Reprinted with permission.

perceived as less well planned, may result in more interference. Although experience does not seem to justify this presumption—indeed Agile audits seem to focus much more on material issues—the bottom line is that management and auditees usually reply positively to success. If Agile audits result in more timely and material results, they will probably not only be accepted, but also preferred. As discussed earlier, the potential is there because Agile audits can save time and, hence, afford more time for material issues. A great deal depends, of course, on the actual execution, which supports the importance of a competent team and leadership. This, however, is no different from traditional audit management.

Figure 2 compares Agile and non-Agile audits.

Of course, whatever audit methodology is selected must be approved by the enterprise.

Agile Audit Guidelines

Although each audit may have its own unique characteristics, some of the Agile audit guidelines include:

- Strive to gain an early understanding of the key audit issues and disseminate this information within the audit team. For instance, what needs to be checked and what will be needed? This dissemination can be in the form of a simple email and need not be formal.
- Request data known to be necessary straightaway, without waiting for team meetings. If getting all data is time-consuming, focus on a sample of the data. For instance, if extracting a large quantity of data is both necessary for the audit and time-consuming, request a few lines immediately and set a time frame for the remaining data. Communicate data as they come in to the audit team. This and the previous task will normally be done by the lead auditor.
- Hold informal meetings with the team and auditees to discuss the issues, ensure that important risk areas identified by the team or the auditees are not left out, verify that all necessary data have been collected or requested, and assign tasks. Keep track of work done, work assigned, points discussed, etc., but not in a formal document.
- Discuss findings as they are gathered. Once they are accepted, document them and, if possible, verify with the auditees. There is no need to wait for all the findings to verify a single, documented finding. Auditees are often more likely to

accommodate a short time to discuss one finding than a much longer time to discuss a number of findings. If a finding is verified, include it in the draft report and update the report as verified findings become available. This way, report writing time is also shortened.

- Shift resources if necessary. If a team member finishes with work, perhaps because the data needed were available first, that team member can be available to aid another member.

Conclusion

Audits, being essentially a project, can employ the highly efficient methods from Agile development for all but compliance audits. These methods are especially appropriate for complex audits and require a team of competent and experienced auditors. Auditors must remember that the bottom line is to add business value. If a methodology serves this end, then it should be embraced. If it is a hindrance, it should be dropped. With rising requirements on internal audit, namely, to provide timely assurance on material issues,[17] Agile methods in audit can be of great help.

Endnotes

1 Agile, “Manifesto for Agile Software Development,” 2001, *http://agilemanifesto.org/*

2 ISACA®, *CISA® Review Manual, 23rd Edition*, USA, 2013, p. 191

3 Srinivasan, S.; *Advanced Perl Programming, First Edition*, O’Reilly Publishing, USA, August 1997

4 Saint, C.; “Can We Make Internal Auditing “Agile”?,” *Internal Auditor*, 2 July 2014, *https://iaonline.theiia.org/can-we-make-internal-auditing-agile*

5 Prickett, R.; “Agile Auditing,” *Audit & Risk*, 10 July 2015, *http://auditandrisk.org.uk/features/agile-auditing*

6 Darlison, T.; “Agile Auditing—What It Means and How to Do It,” presented at IIA Annual Conference, September 2015, *https://www.iia.org.uk/media/1431921/tony-darlison-day-1.pdf*

7 Hancock, B.; “Agile Audit,” The Ohio State University, USA, 31 May 2015, *https://u.osu.edu/auditagile/*

8 Spencer, A.; “Suncorp—Agile and Internal Audit!,” AgileBusinessManagement.org, 3 December 2013, *http://agilebusinessmanagement.org/content/suncorp-%E2%80%93-agile-and-internal-audit*

9 *Ibid*.

10 *Op cit,* Prickett

11 *Op cit,* Darlison

12 *Op cit,* Hancock

13 *Op cit,* Spencer

14 *Op cit,* Prickett

15 Chambers, R.; “Drive-by Auditing: Don’t Be Guilty of ‘Hit and Run,’” Internal Auditor, 2 August 2012, *https://iaonline.theiia.org/drive-by-auditing-dont-be-guilty-of-hit-and-run*

16 Berkowitz, A.; R. Rampell; “Drive-by Audits Have Become Too Common and Too Dangerous,” *The Wall Street Journal*, 9 August 2002, *www.wsj.com/articles/SB1028822538710052160*

17 Marks, N.; “The Agile Internal Audit Department,” Resolver, 2014, http://resolver.com/wp-content/uploads/2014/06/The-agile-internal-audit-department-Norman-Marks-Resolver-2014.pdf