BEST PRACTICES

Building a remarkable Archer team

A successful GRC program is one which evolves with your organization, adapting to your users, your stakeholders, and your industry. With RSA Archer, many organizations spend a lot of time (and resources) planning for the initial implementation, but not enough time considering how to properly support and update their program once it is in production.

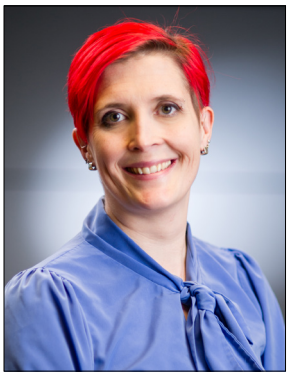
So how do you build a team to properly sustain your Archer program? What skills should you be looking for? How should you develop your resources? Here are some pointers based on our work with Archer customers and our own experience in building Iceberg's team. I've broken it down into four areas:

- **Understanding Archer** – How is it like traditional coding technologies and how does it differ?
- **Ingredients for a valuable Archer resource** – What skills should you look for? Do they need Archer experience?
- **Archer training and mentoring** – Even when a resource is trained in Archer, they need to know the specifics of the implementation. What's the best way to support that?
- **Creating institutional memory** – What are some strategies to create a framework that your team can refer to, and avoid having all that knowledge sit in the mind of a single person?

1. Understanding Archer

RSA Archer is both like and unlike traditional coding technologies. If an application is created in traditional code, like .NET or C#, even a resource who is familiar with those languages requires time and documentation to get up to speed on the intricacies of the application in your specific environment. Without documentation or an understanding of the business processes that contributed to the application's creation, even the most senior developer will need to spend a significant amount of time finding the root cause of a bug, or creating an augmentation for that application.

This is also true of Archer. Depending on your configuration, it can become just as complex and requires just as much skill as managing an application in traditional code. Similarly, you'll need to have governance and controls in place, particularly as implementations mature – although how they are implemented may be different.



By Melissa Cohoe
Director of Implementation & Integration
icebergnetworks.com

On the other hand, Archer is not traditional code. It's a framework we can use through a user interface to design screens, layouts, and workflows, all built on the backbone of out-of-the-box applications designed by risk practitioners. Many configurations are easier to do within Archer's framework than traditional code, while others require more creativity.

What kind of people do you need to work on a tool that is both like and unlike traditional code? You need people who are both like and unlike traditional coders!

What kind of people do we need to work on a tool which is both like and unlike traditional code? You need people who are both like – and unlike – traditional coders!

2. Ingredients for a valuable Archer resource

When looking for resources who can own, grow, and create standards specific to your company, you want resources with the following attributes:

- **Avid Learners** – Risk managment is always an evolving environment. It requires re-evaluation to identify the best way forward, and the ability to reassess and adapt your Archer environment as business needs change. A passion for learning can translate to learning about the business processes, which can only improve your Archer implementation.
- **Logical Thinkers** – Being able to systematically work through a problem is critical for any technology resource. This is frequently a skill you can find in traditional coders.
- **Creative Thinkers** – This is where we go beyond the traditional coder. As important as being systematic is, there are end-user requests that need out-of-the-box thinking. By looking at a requirement a different way, more efficient solutions can be found.
- **Dynamic** – This is the attribute that blends everything together. Being able to react quickly, accept challenges as they are offered, and the ability to think both logically and creatively are all attributes that allow an Archer resource to grow, to be passionate about your business, and to really own their part of the Archer implementation.

Do your resources need Archer experience before you hire them? Not necessarily. The choices are to either commit to taking the time to find someone with Archer experience, or hire someone and commit to training them in Archer. Our approach has been to train our talent at Iceberg in Archer and mentor them on how Archer works. We have not found it to be a significant challenge to ramp up resources when we use the training and practice guidelines outlined below.

3. Archer training and mentoring

The necessity of thorough Archer training and mentoring cannot be overstated. This starts with RSA's formal training and certification program. By learning the Archer basics, nomenclature, and standards, your resources will be better armed to understand both how Archer works, and how the Archer implementation works in your specific environment.

As your resources learn the basics, they'll need somewhere to practice, experiment, and even fail without risk. We recommend giving new resources access to skills development sandboxes where they can develop their skills, and as they become more experienced, really start to grow their understanding of Archer.

SKILLS DEVELOPMENT SANDBOX

Iceberg offers private cloud-based Archer environments that your team can use for prototyping, skills development and exploration. Individual sandboxes can be provisioned for your staff, allowing them to try out configuration changes and data manipulation without affecting official development environments.

icebergnetworks.com/sandbox

Iceberg uses the following stages to assist understanding the expected growth of an Archer resource from an entry level position to an architect position. Senior resources will already have a foundation of experience which they can apply, so they can likely pass through the early gates more quickly.

Entry Level
(~3-6 months)

Is already (or becomes) Archer certified.

Is provided an Archer Sandbox.

Begins to work on smaller, specific ticket requests.

Learns how Archer works and documentation objectives.

Begins to learn high level workflows within the customer environment.

Able to manage small and medium tickets with minimal guidance.

Intermediate
(~6-24 months)

Begins to work on small (<100 hours) to medium (250-500 hours) projects.

Begins in-depth learning in one of the customer environment applications.

Iceberg Boot Camp Phase 1 – Configuration Best Practices, Documentation Best Practices.

Able to manage small projects without minimal guidance.

Lead
(~24-48 months)

Begins to work on large (1,000 hours or more) projects.

Can create efficient calculations and data driven events.

Able to troubleshoot and work on small problems for which they are not familiar.

Able to mentor entry level and intermediate resources.

Familiar with Archer's out-of-the-box modules as well as customer's modified modules.

Iceberg Boot Camp Level 2 – Risk 101, Mentorship, Architectural Standards, User Experience Design.

Able to manage medium projects and guide other resources to the same level.

Architect
(on-going)

Able to manage large projects and the efforts of other developers.

Able to articulate the basic design principles to both end-users and technical resources.

Familiar with all customer applications and confident in their ability to solve problems in areas with which they are not familiar.

Able to perform large scale implementations, creating coherent designs and roadmaps for the future.

Fully independent Archer resource able to lead other resources and grow them to the same level.

4. Creating institutional memory

Like any configured or developed tool, documentation becomes key for long-term success in technical support and growth of applications. Knowledge that rests only within the mind of a single support resource is lost when the resource goes on vacation or leaves the company. It's important to find ways to cement technical knowledge (or at least the path to it) within the institutional memory of the organization.

Project deliverables are not enough, because the purpose of project deliverables (to create a history on what the project did or is in progress of doing) does not directly align with what an Archer support person needs. An Archer support person needs documentation showing to status of the application in its entirety, as well as the high-level business processes that use the applications and solutions.

If you are at the beginning of your Archer implementation, this is easily managed by building in technical user documentation to your implementation project. If a robust environment exists already, time must be invested to build your documentation. While it may be a heavy lift up front, there are obvious long-term benefits: it will simplify transitions to new resources, and help decentralize your support structure to avoid a single point of failure. Even in cases where there is only one Archer support person, the documentation helps in hand-overs, vacation coverage and other extended absences.

Iceberg has developed the following steps to show quick value on creating documentation for a multi-solution environment.

Phase I
(1 month)

Iceberg to use existing information to create a referential repository for customer GRC applications.

Iceberg and customer to identify the first application to begin documentation, by considering the highest value for the least amount of effort.

Iceberg to begin creating a reference document for chosen application current state.

Customer resources given access to Iceberg resources, from SMEs to "hotline" to ask questions and get answers.

Phase II
(months 2-4)

Phase I documentation completed.

Iceberg and customer to identify next candidate for documentation.

Iceberg and customer to begin work on Phase II documentation, leveraging existing project documentation and the template for the previous document.

Customer team invited to participate in Iceberg Lunch and Learns on development and documentation standards.

Customer team invited to participate in Iceberg's "IceFest", a day of Archer innovation.

Phase III
(months 4-7)

Phase I documentation and Phase II documentation completed, and regularly updated based on new project efforts.

Identify next two candidates for documentation.

Iceberg and customer team to work together on Phase IIIa documentation.

Customer team to create Phase IIIb documentation with Iceberg's assistance.

Create governance for documentation review, renewal and cross-training.

Phase IV
(on going)

Complete documentation of customer applications.

Create on-boarding activities for new resources.

Decentralize knowledge by leveraging resources with more experience to learn and expand knowledge into more than one application.

Regular document updates and reviews key to success.

Share your experience

Part of my job at Iceberg over the past few years has been managing our team of GRC developers, so these topics are very much top-of-mind for me. I would love to hear from you about what works, or hasn't worked, for your teams. Please feel free to reach out at **mcohoe@icebergnetworks.com.**