AEM Assets Migration Blueprint

Part I - Planning

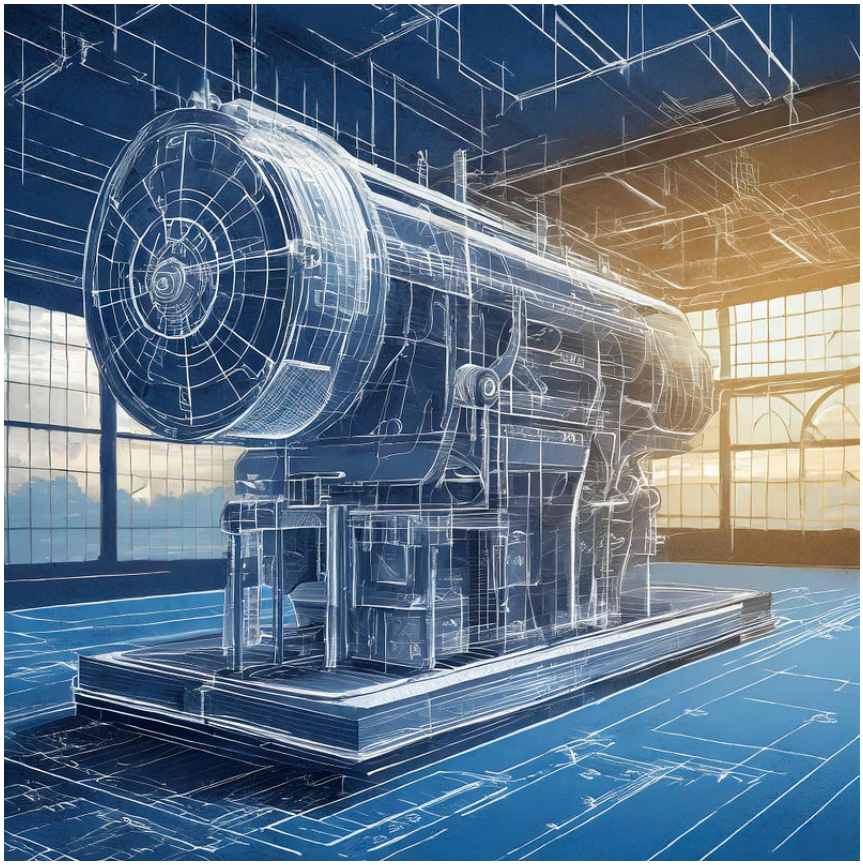


Table of contents

1 Series Overview

In the past, I have done a couple of migrations from arbitrary Digital Asset Management Systems (DAM) to AEM Assets and want to share what I have learned along the way, so you can avoid the common pitfalls.

This is Part 1 of a three parts series on migrating to AEM Assets:

- Part 1 — Planning
- Part 2 — Transforming and tools
- Part 3 — How to not Migrate Taxonomies to AEM Assets

In this part I'll touch upon planning — the aspects of stakeholder management and timing.

The second part will be dedicated to transforming — how to map artifacts from the legacy system to AEM Assets. I'll conclude with a selection of proper tools. Some will be obvious choices; on others I'll share some subjective opinion and experiences.

2 Foreword

Let me start with the presumably "boring" stuff — planning. A migration often is considered as a pure technical exercise and left to developers and architects. The project-management side of things often is neglected.

If you are a developer, feel free forward this article to your project manager and subscribe to be notified about the next article. Though I encourage you to read on anyway — it'll cost you only 10 minutes — and will broaden your horizon. Also, technical decisions often heavily rely on project context — like budget or timing.

Each design decision is a trade-off…

3 Stakeholder Analysis

For a migration you need to consider the several typical stakeholders. Make sure to ensure sufficient availability of the stakeholders throughout the project and to establish stable communication channels with all. This is easier said than done. Often your time is dedicated to the migration project while other stakeholders still have their daily business to conduct.

The typical roles are:

- Project owner
- Project manager
- Business owners
- Business users
- IT operation
- Legacy support and maintenance
- Migration architect / developer

The project owner is the person on the business side who sponsors the project and *oversees* making decisions.

The project manager is responsible for planning time, resources, and budget. He *facilitates* meetings where decisions are being made. A project manager typically does not have deep technical knowledge — though some basic understanding helps. The project manager can be an in-house role or an external consultant.

The business owner is supposed to have in-depth knowledge of the business side of the system and project and a strong mandate to ultimately *make* decisions.

Sometimes, business owners are stewards for a system and not working with the system on a day-by-day base themselves. They might require support from actual business users using the system. The users are supposed to work with the legacy system every day and know the ins and outs, quirks, and workarounds.

Note, these are *roles*. In smaller projects a single person can hold these roles.

IT Operations — These are the people responsible for keeping the legacy system up and running. They are responsible to grant access to the interfaces to facilitate extracting the data. They are not supposed to have the technical in-depth knowledge required for a migration. They operate at infrastructure level and are concerned about business-as-usual-tasks, only.

This is where the Legacy support and maintenance staff comes into play: They are experts for the legacy system and required to support the extraction of the data from the legacy system. They are supposed to answer questions about which interfaces the legacy system supports and how exported data can be interpreted. They could also be tasked to configure or implement export procedures on the legacy system if none exists. Usually, that level of support is provided by external consultants, as this level of detail is not required for business-as-usual after the initial project finished.

Migration architect / developer: This is probably you: The person responsible to conduct the actual migration from the legacy system to AEM. You don't have to have in-depth knowledge of the legacy system — but you need to be able to gather the technical details from the legacy support. You must be able to understand (and sometimes reverse-engineer) the business requirements of the legacy system and help devise the requirements for the new system.

4 Schedule and Deadlines

First, find out the compulsory timing constraints of the projects. For example, the migration might be necessary, because the license contract for the existing system will not be extended or the projected growth of the volume of assets exceeds the capacity of the legacy system.

I have also seen case where the existing system was discontinued and shut down by the vendor.

5 “Do not Touch” Periods

Each business has critical times in the year when changes to systems are considered undesirably risky. They will have a policy forbidding changes during these periods. This can be the launch of an important campaign, the peak season (e.g. before Christmas holiday) or a trade fair. Make sure you know these periods and plan the transition accordingly.

6 Business Continuity and Transition Plan

Created with Adobe Firefly

When *migrating* a system, keep in mind, there always is an *existing* system running and *in use*. Thus, you'll have to plan, when and how to transition from one system to the other — respecting business continuity: Business stakeholders need to be able to work with the legacy system — until the migration is completed and they can seamlessly switch to AEM.

In the past, I have seen a few alternative strategies to provide a seamless experience:

- *Big Bang migration*
- *Parallel operation*
- *Segmented migration*
- *Delta migration*
- *Continuous migration*

You might have guessed it: Where there are options, there are tradeoffs. Otherwise, you would just pick the "best" solution, right?

Let me explain:

Big Bang means, you declare or a enforce a *Content Freeze* on the old system. At some point in time, users are not allowed to upload any new assets to the legacy system. You'll export the data at that point, transform it and ingest into AEM.

This kind of migration is the least complex one and thus the most desirable one. However, a Big Bang means migrating a large amount of content at once. Remember, during that time, we'll have a content freeze: No new assets can be added to the system. Depending on the volume of data this can take a couple of days. Try to plan the transition over a long weekend or during off-peak times, when uploading new assets is not required. For smaller volumes, a "Big Bang" migration is a no-brainer.

To mitigate the downsides of the content freeze there often is a time span of parallel operation during migration: Content is extracted from the legacy system, and users *should not* upload any longer to the legacy system. If necessary, the upload needs to be performed on both the legacy and the new system.

If the DAM is the main ingestion system, it is the responsibility of the business users to do the same on the new system in parallel. If the ingestion happens on another system up-stream, you might consider distributing the assets to the legacy system and AEM in parallel. This requires a bit more coding — but would be less tedious for the business users.

In a **segmented migration**, we try to identify chunks of data that are independent of each other. This could be assets belonging to different departments in a larger enterprise DAM. We would then migrate segment by segment, where each segment would require only a smaller content freeze that could be declared overnight. This requires - of course - that the data can be segmented in the first place. The downside is, this requires more planning — and the overall transition will take longer than pushing through in one big bang. This might be the only option for huge DAMs.

In a **delta migration**, we consider the assets as a whole again. We would do an initial "seeding" big bang migration during business hours without a content freeze. In the later - actual - transition, we would only migrate what has changed between the seed migration and the current point in time. This results in a much smaller change set. During that longer transition phase, the legacy system is considered the leading *system of records (SoR)*. At some point you simply declare AEM to be the SoR and you are done.

At first glance, Delta migration seems to be the most desirable strategy. But there is a catch: It's hard to implement: You need to a track changes on the legacy system since the last delta, and you need to cross-reference and retrace the changes on the new system. That's simple if you are only adding data to the legacy system but becomes difficult when the change is of different nature, such as:

- modifying an existing asset's binary
- modifying meta data
- moving assets
- deleting assets

Adding and modifying assets is more easily to be handled. But if assets can also be deleted or moved on the legacy system, it is difficult to retrace on the new one.

If you can enforce a "soft" content freeze where these operations can be ruled out or denied, then a delta migration can be a viable option. If not, be prepared to write some custom code that is able to cross-reference data between the legacy and the new system.

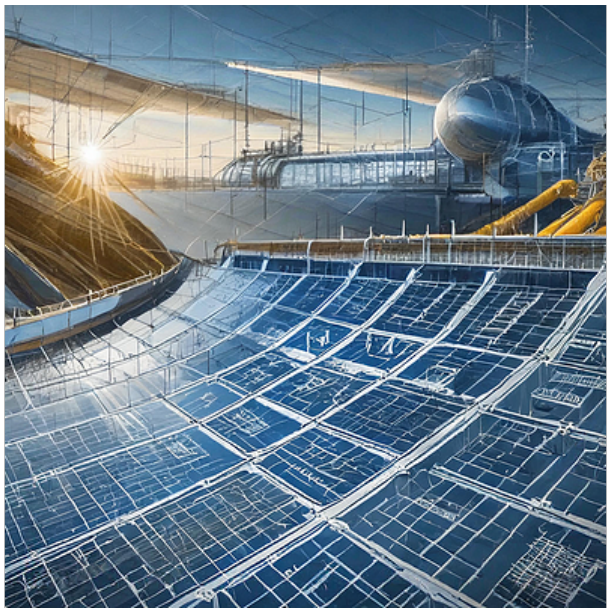
It goes without saying, that during the delta migration, changes to the data on the new system must be avoided:

Example:

If there is a new asset on AEM but not on the legacy system, this can be interpreted as "deleted from legacy" and thus deleted on AEM by your algorithm. If parallel editing is required, make sure to tag and ignore these assets during migration.

Finally: A **Continuous migration** is a delta migration that runs - well - continuously. Deltas are calculated on each change and the new system is updated immediately. This might require changes on the old system. This is something you'd usually avoid as this is considered as sunken costs. The old system will be decommissioned…

7 Infrastructure



The available infrastructure has quite an impact on both the planning and the technical setup. Make sure to understand what infrastructure is available and what you need — and where to get the type of infrastructure that is missing.

Examples:

- If you have 100 TByte of assets on the legacy system but you can extract at a rate of 1 Gbit/sec, only, the extraction will take ~200 hours — not likely to do that in a big bang.
- If the legacy system only allows to pull data from the database, you require some cloud server as "driver" to actively pull the data before transformation. You don't want to do that *in situ* in AEM and you don't want to do that on your local laptop, either.
- If the legacy system is capable of exporting to an intermediate AWS or Azure cloud storage or FTP server — you need to have that storage provisioned (and paid for). While technically a no brainer, this can be a challenge business-wise. Hopefully, this was factored in the quote of the agency conducting the migration … or can be provision by the customer's IT ops division.

8 Conclusion

This was my little crash course in project planning. I bet in real life you'll find more obstacles and intricacies. Please, comment what impediments you have encountered in your project.

With this out of the way, we'll can now focus on the fun stuff — the technical migration. Stay tuned. Follow me to get notified.

Cheers

-achim