Alfresco Communityによる文書管理システムの構築

中村　京子

自然科学研究機構国立天文台 アルマプロジェクト

１．はじめに

オープンソースのコンテンツ管理システムに「Alfresco」というものがある。これには無料版のCommunity[1]が存在し、今回、これを使ってプロジェクトの文書管理システムを構築したので報告する。Alfresco は世界中の公的機関や主要企業で採用されており、文書管理システムとしての完成度が高く、無料版でもさほど遜色はない。ただ、構築にあたっては日本語の資料が少なく情報収集の点で苦労したため、Alfrescoについての知見を広めたいと思い、本研究会で紹介することとした。

２．背景

国立天文台のアルマプロジェクト[2]は、米欧と共同で国際プロジェクト「ALMA」を進めている。国際プロジェクトの本部「Joint ALMA Observatory（JAO）[3]」は南米チリの首都サンティアゴに置かれ、チリ北部のアタカマ砂漠に設置された 66 台の電波望遠鏡を運用している。本台プロジェクトは相関器、分光器のほか、望遠鏡と受信機の一部を ALMA に提供しているため、それらの開発や保守も担っている。なお ALMA には台湾や韓国など東アジアの国々も参加しており、本台プロジェクトはそれらをまとめる東アジア地域センターとしての側面も持つ。日々の天体観測で得られたデータは日米欧の各センターで手分けして一時処理しており、東アジアにおいては、本台プロジェクトがその管理を行っている。

ALMA は計画や検討、設計、実装作業、試験、管理等々で様々な文書を作成する。そのため発足当初から、文書管理システムを設けてそれらを保持している。本台プロジェクトも ALMA に提出する文書はそれを使用するが、日本での開発文書や本台プロジェクト内部の取り決めに関する文書などは、我々自身で管理する必要がある。そこで、市販の ArcSuite（Fuji Xerox）、DocuShare（Xeros）等のほか、Procenter（NEC）、楽々DocumentPlus（住友重工）、JIRA など様々な製品について調査をしたのだが、検討が長期に渡ったにも関わらず最終的な決定に至らないどころか、一時頓挫もした。そのため、様々な文書が複数の場所で管理されるという状態が長く続いた。決定できなかった理由は、文書管理システムに求める機能を最初から高くしていたためであった。例をあげれば、レビュープロセスに対応するワークフローを備え、レビュアのコメントを随時書込め、権限のある者はそれらを閲覧でき、リンクされた文書の編集も誰がいつという情報も含めて可能、といったものである。費用をかければ問題ないだろうが、本台の場合は予算が限られていたため、頓挫もやむをえない面があった。

ところで ALMA 自体は長期プロジェクトであり、発足からすでに 20 年経過しているものの機能の高度化を現在進めており、まだ数十年は続くことが見込まれている。そのため我々独自の文書管理システムの設置が焦眉の急であった。そんな中、JAO が文書管理の次期システムとして Alfresco の採用を決定した。JAO は有料版を使うが、Alfresco には無料版があるため、本台プロジェクトでも Alfresco 選択に踏みきった。

３．文書管理システムの構築

３.１　システム決定から運用までの全体の流れ

文書管理システムとしてAlfrescoが決定されてからのおおまかな流れは以下の通りである。

2020年9月 本台プロジェクトの文書管理システムとしてAlfresco Community採用決定
2020年12月 JAO既存文書管理システムの、東アジア作成分文書の移行支援
2021年4月 文書管理システム検討開始
2021年8月 要件定義書策定
2021年9月 設計作業開始（6名でチーム発足）
2021年11月 設計レビュー会開催、設計書完成
2021年12月 文書管理システム運用機構築開始

2022 年 1 月 既存文書移行
2022 年 2 月 文書管理システム運用開始

4 月に検討を開始してから 10 ヵ月という短期間で運用までこぎつけられたのは、Alfresco 自体が完成されたシステムだということが大きい。さらに、ステークホルダやヘビーユーザに聞き取り調査を行いシステムに求めるものを把握した上で、今までの失敗を踏まえ、段階的に機能を強化していく方針としたことも効いた。具体的には以下の三つのフェーズを設けた。2022 年 2 月からの運用は、第一段階のものである。

【第一段階】基幹部分の構築。散在文書を一つのアーカイブにまとめて文書の一元管理を実現する。
【第二段階】機能追加 1。作業効率を強化する機能とレビューワークフローを実装する。
【第三段階】機能追加 2。第二段階で見送った、ユーザ要望の多い機能を実装する。

３.２　構築作業

ユーザの範囲や取り扱う文書の種類、ユーザ権限、アクセス方法等の検討と同時進行の形で、試験機上で Alfresco の動作を確認した。Alfresco は仮想マシンで動作させることとし、OS は RHEL8 とした。検討に基づいて設計書を作成、レビュー会を実施し、設計書を書き上げた。ただ十分な時間がとれなかったため、試験機ではすべてを確認することができず、少なからずの機能は運用機の構築時に確認した。

そのうちの一つがインストール方法だった。Alfresco のインストールには三つの方法がある。パッケージを取得して手動で設定する「Install with zip」、自動インストールツールを使う「Install with Ansible」、Docker で構築する「Install using containers」である。カスタマイズしやすいのは手動設定だったため Install with zip で始めたのだが、検索機能が動かない。そこで Ansible に方針転換したが、自動インストール途中でエラー終了してしまい、Ansible には不慣れですぐに解決できなかったこともありこの方法は断念した。Docker 利用は滞りなく構築でき、Alfresco の動作も問題なかったのだが、手動設定に比べてカスタマイズに手間がかかり、結局は不採用とした。ところで、Alfresco は本体機能・検索機能・レンダリング機能の三つの大きなパッケージ、データベース、そして複数のツールで構成されている。そのためバージョンが合っていないと各々の連携がうまく働かない。そこでもう一度それらのバージョンを精査し、Install with zip でやり直したところ、無事、すべての機能が使えるようになった。

以上、少々手間取りはしたが構築作業は終了した。

３.３　文書移行

我々の文書管理システムは、プロジェクト長により「NAOJ ALMA EDMS」と命名された。設計書に合わせてフォルダツリーを変更し、権限設定、Alfresco 機能の制限などを行った後、既存文書を移行した。先に述べた通り従来の文書の置き場は様々で、大きな所でも自作の簡易システム、ネットワーク上の共有ディスク、「サイボウス」の三つがあった。そこで少なくともこれらを新システムに移行した上で、運用を始めることになった。

Alfresco には「Bulk Import」というツールが備わっており、これにより大量文書をメタデータを含めて一気に取り込むことができる。そこでオンラインで文書を集め、文書名・作成者・作成年月日・版数・グループ名・機密レベル等のメタデータファイルを文書ごとに作り（これは専用ツールを自作した）、Bulk Import で新システムに移行した。用意のできたものから順次インポートしたため、全文書を一気にインポートした場合の所要時間は不明だが、例えば数百の文書でも数分でインポートは完了した。以上でシステム側の準備が整い、2022 年 2 月 1 日に運用を開始した。

４．運用開始後のトラブル対応

４.１　ディスク圧迫

運用が始まった後、まずディスク容量が圧迫されるという現象に何度か遭遇した。

（１）/ の使用量が 100%

原因は設計ミスにある。インストールの際、Alfresco 本体パッケージをルートディレクトリに置いたのだが、そのサブディレクトリに文書が保管されるということを失念していた。そこで、大容量パ

ーティションに文書を置くこととし、本体側にリンクを張ることで解決した。

（2）Alfresco ログの肥大化

これは Alfresco の仕様であった。Alfresco のログは、アップロード・削除・編集等の文書操作がないと自動的に肥大するそうである。当時いろいろ調べたのだが回避策はなく、対処療法として定期的に文書操作をするしかない。ただ、運用機については毎日、誰かしらが操作するのでこの問題は生じない。試験機はそうではないため、定期的に文書操作をすることで解決した。

（3）検索インデックスの肥大化

こちらは検索機能の仕様のようである。検索機能は Apache Solr で、Alfresco が独自にカスタマイズしたものを使っている。Solr は管理文書の量が多いとインデックスも大きくなるようで、それに対応するにはインデックスファイルの置き場を変えるしかない。そこで（1）と同じやり方で解決した。

4.2 メモリ圧迫

運用を始めて一年近く経った頃、検索機能を実行しているプロセスが自然消滅したことがあった。調べたところ、Solr が OOM Killer（Out Of Memory Killer）により実行プロセスを終了させたことがわかった。Solr は Java で動いており、その最大ヒープサイズが不足していたことが原因だったため、メモリを 8GB から 12GB に増やし、Solr が利用する Java の最大ヒープサイズを 1GB から 2GB に増やして解決した。

なお、後に文書管理システムをアップグレードした際、検索インデックスファイルが以前より大きくなり、再び OOM Killer が発生した。メモリにまだ空きはあったため、Java 最大ヒープサイズのみ 2GB から 3GB に増やして対処した。このように我々の場合は仮想マシンのためメモリ増強は容易だったが、そうでない場合は、文書量を予測して最大ヒープサイズがどれくらい必要かをあらかじめ見積もっておく必要があるだろう。

4.3 LDAP エラー

これはプロパティファイルの設定ミスになる。Alfresco はローカルユーザを作成できるが、Active Directory（AD）と連携し、AD アカウントでログインすることも可能である。本台プロジェクトでは別システムで AD を運用しており、文書管理システムでもそのアカウントでログインできるよう連携設定していた。ログイン自体は問題なかったのだが、Alfresco は定期的に AD に問合せをしており、そこでエラーが発生していた。AD のルート DN の指定が間違っていたことが原因だったため、正しい DN を Alfresco のプロパティファイルに書込み、Alfresco を再起動することで問題は解決した。

5. 保守作業

5.1 環境設定

運用開始までは自転車操業だったこともあり、十分な環境整備ができていなかった。そのため、運用が始まった後で各種整備を行った。

まず、自動起動と停止を実装した。スクリプトを作成し、本体システム起動時に文書管理システムが自動起動し、停止時には自動停止するようにした。続いて、ログローテーションの設定と、cron による Alfresco ログの定期移動を実装した。

5.2 脆弱性対応

Alfresco は Apache Tomcat を利用している。運用開始後、脆弱性対策のための Tomcat のバージョン更新を年に数回の頻度で行った。その他、この間に Apache Active MQ の脆弱性報告があったため、これのバージョン更新も実施した。

6. まとめ

以上で Alfresco Community を使った文書管理システム構築についての報告を終わる。実の所、まだ第二段階へは進んでいない。システムの運用開始後、アルマプロジェクト内の人の割り当てが変更され、第一段階の時とは働き手の環境が変わってしまったことが原因である。現在も見通しが立っておらず不本意に思っている。

一方で、人手不足の中、文書管理システムとしてそれなりのものを立ち上げられたのは、Alfresco 自身の完成度の高さに負うところが大きい。さほど手を入れなくとも基本的な機能が十分であり、構

築時の負担が大幅に軽減された。Alfresco は英語文献が多いことからも日本ではあまり広がっていない印象を受けるが、製品自体はとても優れているので、コンテンツ管理システムを検討しているのであれば、候補の一つとして取り上げることをお勧めする。

参考文献

[1] Alfresco Community URL https://docs.alfresco.com/content-services/community

[2] アルマプロジェクト URL https://alma-telescope.jp/naoj/

[3] JAO URL https://www.almaobservatory.org