SWORD v2.0: デポジットのライフサイクル

SWORD は大成功を収めた JISC プロジェクトである。リポジトリの相互運用性を大きく高め、リポジトリとその相互運用性の問題に関する新たなコミュニティを構築した。SWORD は明らかにパッケージデポジットによるリポジトリリソースの新規作成という簡単なケースのみを扱っている。それは SWORD が成功した要因であるが、一方でその限界の大きな要因にもなっている。SWORD の次期バージョンは、仕様に更新、検索、削除という拡張機能を追加することにより、リポジトリに対するデポジットのライフサイクル全体をサポートする方向にこの標準を推し進めるものである。これにより、ますます広がる様々な操作やユースケースに対応でき、学術環境における幅広いシステムにリポジトリを統合することが可能になるだろう。

***著者 / SWORD v2.0 リリース責任者**: Richard Jones, Symplectic 社*
***助成機関**: UKOLN, JISC*
***協力者**: Jim Downing, David Flanders, Graham Klyne, Stuart Lewis, Adrian Stevenson, Paul Walk*

はじめに

SWORD [1] は、2007 年の登場以来、非常に大きな注目を集めている JISC [2] プロジェクトであり、リポジトリコミュニティに多大なインパクトを与えてきた。SWORD は、どのようにしてリポジトリに相互運用性をもたせるか、また、どのようにして他の情報システムにリポジトリを埋め込むかといった課題に取り組むために始められた。

これは簡単な課題ではないが、SWORD は、その実現が最も容易ではあるが実際的な価値を有するユースケースを扱うことにより、この課題に取り組むことを選択した。すなわち、ファイルとメタデータの双方を含むコンテンツパッケージをリポジトリシステムへデポジットし、受領通知でデポジットの承認を行う方法である。

これを実現するために、SWORD はさらに次の点を考慮する必要があった。

- リポジトリは通常、コンテンツを「コレクション」としてまとめており、デポジットされるコンテンツはこれに投入されなければならないという事実 [SWORD 仕様 B.8 節]
- パッケージのフォーマットと内容をリポジトリに示す方法 [SWORD 仕様 B.8 節]
- 認証と承認。具体的に言えば「On Behalf Of」デポジット（代理投稿）ユースケースのサポート [SWORD 仕様 A.2 節]

その結果、相互運用性への道に沿ってこの第一歩を踏み出すためには大量の作業をこなす必要があった。

オープンスタンダードとオープンソースのソフトウェア開発、さらに仕様に対する包括的アプローチを採用したことで、SWORD に関するコミュニティが構築されることになった。SWORD にはオープンソースのクライアント、サーバ実装が存在する。DSpace [3]、EPrints [4]、Fedora [5]、arXiv [6] は SWORD をサポートしており、Zentity [7] や IntraLibrary [8] といった数多くの商用システムも同様である。Facebook にはデポジットアプリケーションが存在する [9]。Microsoft は文章をアーカイブに投稿する Word のアドオンを開発しており [10]、Open Journal System（OJS）も同様な機能を持っている [11]。

リポジトリはもはや、独自のアーキテクチャ選択に依存した独立した存在ではなく、相互接続されて、ますます複雑になっている学術基盤システム環境の一部となっている。この環境で自らの役割を適切に果たすために、リポジトリは洗練された適切な API を提供することにより、接続先のシステムが自らのサービスを利用できるようにする必要がある。これこそ SWORD が果たすべき役割である。しかし、現状ではいくつかの限界がある。以下では、次に手を打たなければならない点を知るためにその限界を列挙する。

SWORD の主な限界

1. リポジトリの作業の一部を外部システムが行わなければならない

SWORD は、コンテンツとメタデータを一つにまとめ完全に準備してからリポジトリにデポジットするよう要求しており、その負担はクライアントシステムに負わされている。そのため、クライアントシステムはリポジトリにデポジットする前に構造化されたファイルとメタデータを保管・管理することができなければならない。リポジトリはまさに構造化されたファイルとメタデータを保管・管理するように設計されているので、これでは努力の重複である。この環境においてリポジトリが果たしうる役割は、パッケージ化されたコンテンツセットの単なる終着点になることではなく、それこそコンテンツを保管するサービスプロバイダになることであるという強い意見が存在する。

2. アイテムがいつアーカイブできるかをユーザが知らなければならない

第 1 の限界の結果として、SWORD はさらに、ある時点のファイルとメタデータがアーカイブに保管されるべき「完成品」であることを表明する責任を負うことをクライアントシステムのみに求めている。そのため、クライアントは、パッケージ化してリポジトリに配送するために、ある時点でコンテンツに区切りを付ける必要がある。この判断は難しい場合が多く（たとえば、論文出版のライフサイクルのどの時点でアーカイブされるべきかなど）、不可能な場合さえもある（継続的にデータセットが作成されている場合など）。リポジトリがコンテンツ保管のためのサービスプロバイダであれば、まさにいつアーカイブされるべきかを決定する責任をユーザと、アーカイブに対するスキルがより高いと思われるリポジトリ管理者の間で調整することができるだろう。個別のリポジトリにおいて「アーカイブ」という概念が何を意味するかについてはここでは明言しない。真のアーカイブの場合もあれば、オープンアクセスインターフェースによる単なる再発行の場合もあると思われる。

3. SWORD 用の AtomPub プロファイルが明確でない

実用的な観点から言えば、SWORD は基本となる AtomPub [12] のすべての機能を使用するための明確なガイドラインを提供していない。AtomPub の機能をすべて実現するサーバを実装することも可能かもしれないが、今のところ、助成による SWORD 実装にそのような例は存在しない。さらに、SWORD が AtomPub のデポジット（HTTP POST）機能に拡張やプロファイルを追加したことを考えると、他の機能にも同じことが必要になると思わ

れる。たとえば、SWORD は新たな HTTP ヘッダを追加することにより“On Behalf Of”（代理投稿）ユースケースをサポートした（[SWORD 仕様 A.2 節]）が、HTTP POST の使用によるデポジット以外のケースでこれらのヘッダをどのように使用すべきかについては何も述べていない。

4. 構造化パッケージに依存している

SWORD はペイロードの配送には構造化パッケージに完全に依存している。これは相互運用性に係わる重大な問題に目をつぶったものであるが、それは第一歩を踏み出すための意図的なものである。パッケージ化されたコンテンツを確実に識別することは困難である。パッケージには、コンテナの MIME タイプ（たとえば、application/zip）やパッケージの内部構造、マニフェスト文書のフォーマット、構造化されたメタデータや書誌メタデータなど時には多段階の入れ子構造にもなる情報格納層（たとえば、METS [13] を考えて欲しい）が存在するからである。短期的にこの問題を緩和するためには、標準的な SWORD パッケージフォーマットを選択することが考えられる。また、いくつかの問題を大幅に軽減できると思われる Atom [14] でサポートされている単一のコンテンツファイルのデポジットや AtomPub Multipart 仕様 [15] によるフォーマット化されたメタデータ文書のサポートも検討できるだろう。

効果

これらの限界を緩和あるいは除去する利益は甚大であろう。*首尾一貫して完全に機能するコンテンツの保管・検索サービス*を提供することにより、ユーザが直接利用する、あるいは管理者が利用するシステムにリポジトリを*統合するのが容易*になるだろう。作成された*初期の段階からコンテンツにアクセスする*機会をリポジトリとリポジトリ管理者に提供し、保存と公開という面でのリポジトリの任務を容易にする機会も提供するだろう。さらに、ユーザが直接利用するシステムにおいては、ファイル管理用の簡単な*インターフェース*を提供することにより、*リポジトリの詳細を利用者から隠す*ことが容易になるだろう。**長期的には、限界の緩和や除去により、リポジトリがパートナーである他のシステムにサービスを提供することが容易になれば、リポジトリのコンテンツや利用が増え、その結果、リポジトリの所有者やコミュニティ全般に対する価値が増加することが期待される。**

上で述べたすべてを処理する機能を直ちに導入することは不可能である。そのため、SWORD の次期バージョンでは、デポジットのライフサイクル全体をサポートする道を開くと思われる 1 つのきわめて重要な分野に限定する。すなわち、デポジットの受領通知でクライアントに提供される URI と URI が指定するリソースである。

識別子

現在、SWORD の受領通知では、以下の識別子が規定されている[SWORD 仕様 B.9.6 節]。各識別子の採りうる解釈と共に示す。

1. atom:content 要素 。これは、次のいずれかの URI を提供する 。
   i. リポジトリアイテムのスプラッシュページ
   ii. リポジトリアイテムの機械可読の記述
   iii. デポジットされたオリジナルパッケージ
   iv. (i)と(ii)の組み合わせ（たとえば、スプラッシュページに埋め込まれた RDFa など）
   v. その他の特定されない識別子

2. @rel="edit-media" を属性に持つ atom:link 要素。これは、次のいずれかの URI を提供する。
   i. HTTP PUT により更新可能な、あるバージョンのリソース
   ii. パッケージからメタデータと解凍されたバイナリファイルへのアクセスを提供する表現。

3. リポジトリ固有の @rel 属性値を持つ任意の数の atom:link 要素。これには、解凍されたアイテムの構成部品の URI を含めることができる。

4. @rel="edit" を属性に持つ atom:link 要素。これは、次のいずれかの URI を提供する 。
   i. デポジット受領通知と同等の文書を得ることができ、HTTP DELETE リクエストの発行対象にもなるエントリ文書。
   ii. ジャンプオフ・ページ
   iii. ユーザのプライベートメタデータ
   iv. ワークフロー管理ページ
   v. その他の特定されない識別子

これが示すように、リポジトリが使用すると思われる数多くの用途を考慮して、仕様は通知される URI のいずれに関しても特に明確な規定をしていない。これは興味深いことである。なぜなら、正式にサポートする方法を検討する際の要件を暗示しているからである。まず、これら URI の各々の意味を AtomPub [AtomPub 仕様 9.6 節] に沿って提示しなければならない。

1. atom:content 要素は、メディアリソースを指定する。パッケージデポジットの場合、メディアリソースとはパッケージそのものである。したがって、この要素はデポジットされたオリジナルパッケージを入手することができる URI を提供する。

2. @rel="edit-media" 属性を持つ atom:link 要素は、メディアリソースを指定する。これは(1)で述べたようにオリジナルパッケージであるので、HTTP GET の場合、この URI は atom:content 要素と同じコンテンツにリゾルブされるべきである。しかし、新たなパッケージをデポジットする HTTP PUT もサポートできなければならない。したがって、サーバは実装の都合により、このリンクに atom:content 要素に使用された URI とは異なる URI を自由に使用できることを意味する。

3. @rel="edit" 属性を持つ atom:link 要素は、エントリ文書を指定する。パッケージデポジットの場合、これは次のようにふるまうことになる。
   i. HTTP GET の場合、サーバにより作成されるデポジット受領通知と同等な Atom エントリ文書を返す。
   ii. HTTP PUT の場合、定義されたふるまいを持たない。これは伝統的にアイテムメタデータの更新に使用されていたが、パッケージデポジットは独立したメタデータを持たないので、意味を持たないからである。
   iii. HTTP DELETE の場合、オリジナルパッケージや解凍された関連コンテンツなど、リポジトリアイテムそのものの削除を要求する。

これらに加えて、我々の環境において適切なセマンティクスを持つ次のようなリンクを新たに導入するべきである。

4. デポジットアイテムに関するリポジトリ固有の追加情報を記載するために使用する新たな文書を指定す

るリンクを提供する要素。このマニフェスト文書については次節で説明する。

5. リポジトリアイテムにスプラッシュページが存在する場合、このページへのリンクを提供する要素。このページには組み込み RDFa が含まれている場合もある。

これらの新たに追加するリンクが Atom においてどのようにシリアル化されるべきであるかについては規定していないことに注意されたい。これには 2 つの方法が考えられる。1 つは、SWORD 固有の @rel 属性値を持つ atom:link 要素を使用する方法であり、もう 1 つは、sword:manifest や sword:splash といった新たな独自のマークアップを埋め込む方法である。前者は Atom の標準的なアプローチに沿った方法であるが、新たな @rel 属性値は正式に承認されることが強く推奨されており、これには IANA 登録手続きを通す必要がある [16]。後者は標準として規定することは容易であるが、Atom のアプローチに沿っていない。

また、現行の SWORD 仕様ではあいまいに使用されていた多くの識別子は 4 項で述べたリンクに押し込んだ。これは、現行の仕様では暗示されているすべての関連する追加情報をクライアントに提供できる文書を定義する機会が与えられたことを意味している。

マニフェスト文書

SWORD 次期バージョンに提案しているもう 1 つの大きな追加が「マニフェスト文書」リンクの下に置かれる内容である。リッチなデポジットクライアントは、以下の情報を使用してユーザエクスペリエンスを高めることができるだろう。

1. リポジトリにおけるアイテムの状態

ほとんどのリポジトリは、アイテムが存在しうる状態としておおよそ次の 3 つの状態を持っている（リポジトリソフトウエア DSpace, EPrints, Fedora の分析に基づく）。

- *準備中*: 通常この状態は短期間であり、新たなリポジトリアイテムが存在を開始した状態である。この段階で、ファイルやメタデータがアイテムに追加されるが、一般にその作業は一人のユーザが行う。通常、リポジトリ管理者がアイテムに直接アクセスすることはない。

- *審査中*: リポジトリアイテムの準備が完全に整ったら、通常何らかの審査プロセスにまわされる。リポジトリ管理者はメタデータの検証や著作権処理などの一般的な作業や、アーカイブの妥当性判断などのリポジトリ環境や組織固有の審査を行う。リポジトリや SWORD は幅広く利用され、様々なワークフローが存在するので、審査段階には共通の明確なワークフローは存在しない。

- *アーカイブ済*: 審査が終わるとリポジトリアイテムはアーカイブされる。これは、アイテムに永続的識別子が付与されて一般に公開されることを意味する場合もあれば、保存のために未公開のアーカイブに保管されることを意味する場合もある。アーカイブの目的や結果はここでは扱わない。

さらに、次のような状態も必要だと思われる。

- *削除済*: リポジトリアイテムはある時点ではアーカイブされていたが、その後削除された。

- *リジェクト*: リポジトリアイテムはリジェクトされ、リポジトリにアーカイブされなかった。

将来の SWORD 標準やローカルニーズによる固有の実装において新たな状態を容易に追加できる拡張性を持たせるために、これらの状態を扱い、それをマニフェスト文書でシリアル化するための簡潔なオントロジーを導入することが提案されている。

また、各状態がリポジトリにとって何を意味するかを説明する機会をリポジトリに提供することも望ましい。これは、デポジットの過程でパッケージに実際何が行われたかをリポジトリが説明できる sword:treatment 要素 [SWORD 仕様 B.9.8 節] と同じ考えによるもので、デポジットが今現在どうなっているかを詳しく説明するものである。各状態を URI で表現することで、クライアントが理解できる非標準的な状態の追加も可能になるだろう。

リポジトリがこれらの状態を説明する機能を持つようになれば、現在の撃ちっぱなし方式に比べて、アイテムの進捗情報を提供してデポジットした者が処理状況を把握できるようにするが SWORD クライアントには容易になると思われる。そして、**リッチなクライアントを作成させたり、ユーザにデポジットを行おうとする気を起させる可能性が増加するだろう。**

手短に言えば、マニフェスト文書には次の 2 つの要素を追加したいと考えている。

1. アイテムの状態に対する表明（準備中、審査中、アーカイブ済、など）。
2. リポジトリにおけるこれら状態の意味を説明する記述を指定する識別子。

さらに、本文書は、この標準の方向性に関する議論を生み出すことを意図しているので、必ずしもリポジトリでないシステムにも役に立つ共通の一般的な状態も探している。

2. リポジトリにおける解凍されたオブジェクトに関する情報

多くのリポジトリは、投稿されたコンテンツを解凍して、個々の本来のフォーマットでインポートするものと思われる。通常、投稿コンテンツは次のようなものを含んでいる。

- トップレベル・オブジェクト: 事実上、これはパッケージに相当し、すべてのコンテンツが格納されるコンテナである。

- 関連ファイル: パッケージから個々のファイルとして抽出されたアイテムの実際のコンテンツ。フォーマット変換やサムネイルなどリポジトリによって作成された追加ファイルを含む場合もある。

- メタデータ: アイテムに関する構造、管理、書誌などのあらゆるメタデータ で、パッケージフォーマットにより、メタデータファイルとしてパッケージ化されて提供される場合もあれば、マニフェスト文書自体に埋め込まれている場合もある。関連ファイルと同様に、オリジナルパッケージに含まれれていないメタデータを含む場合もある。

ユーザにこの情報を提供できることは、リッチなデポジット環境を構築するという意味でクライアントは関心を持つだろう。ファイルにさらなる操作を行えるように個々のファイルの識別子を含めることももちろん可能である。たとえば、HTTP DELETE でクライアントが個々のファイルを削除できるようにしたり、HTTP PUT であるファイルを別のファイルに置き換えたりすることができるだろう。

アイテムの記述にメタデータを含めることは大きな課題であるので、ここでは扱わない。これに関する今後の作業としては、WebDAV の PROPFIND や PROPPATCH と同等なオペレーションを出発点として検討したり [17]、リポジトリアイテムを表すエントリ文書に対して HTTP PUT を実行することが考えられる。ただし、これらのアプローチがどの程度有効で成功するかは明らかではない。

したがって、本文書では、解凍されたアイテムの基本的な構造に関する情報を何らかの一般的なシリアル化方式で公開すること、ただし、実際のメタデータコンテンツの処理には一切触れないことを勧告する。つまり、コンテンツファイルのみに焦点を絞ることになる。

概要

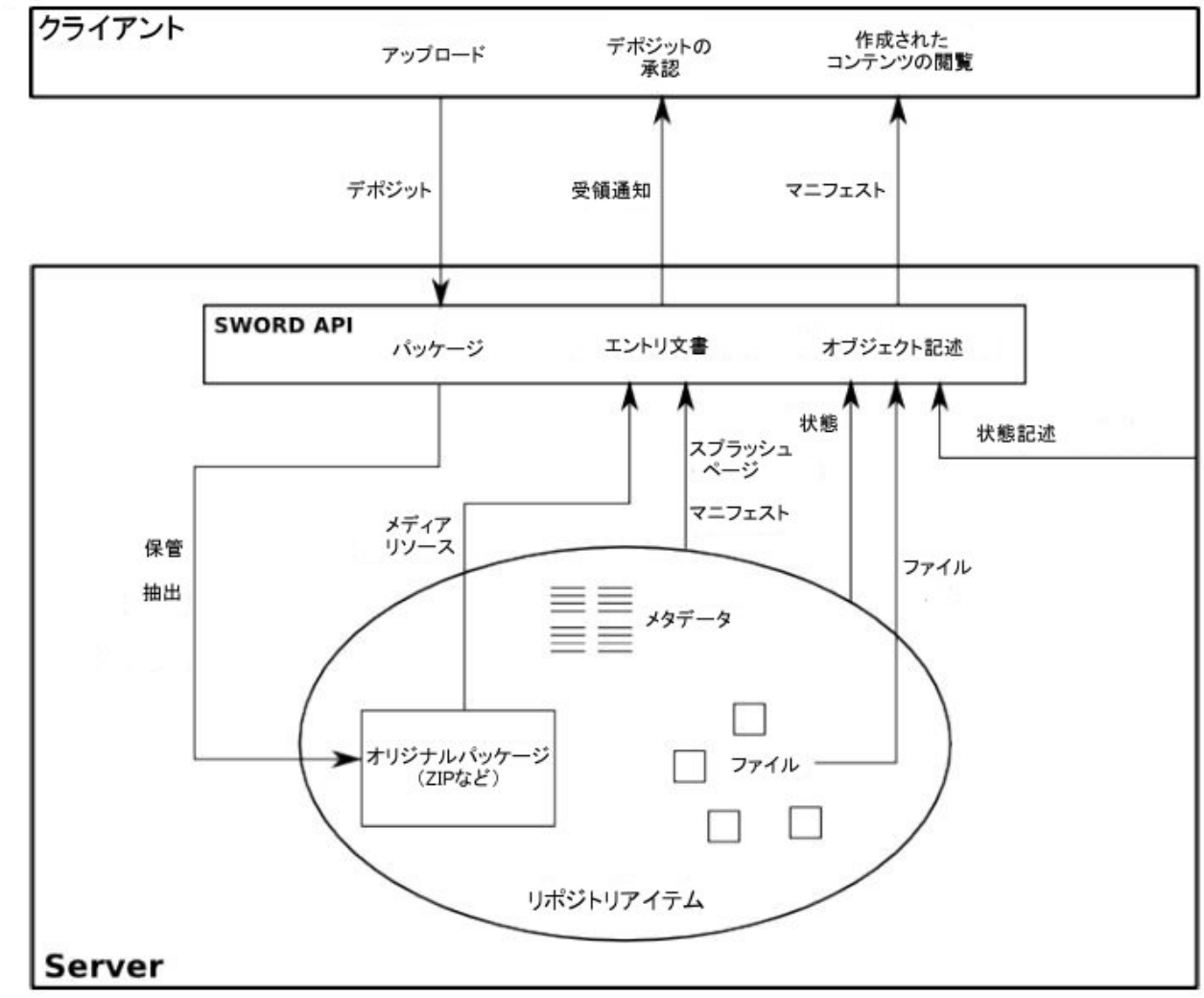

図 1: SWORD 2.0 仕様における追加項目の概要

仕様の拡張

SWORD 3 プロジェクトの最大の成果は、次のようにいくつかの小さな仕様に分割することで以前の版を改良した新しい仕様の提案である。

- ATOM および AtomPub の拡張
- HTTP の拡張
- パッケージ化
- これら個々の仕様（HTTP や AtomPub など）を 1 つの SWORD 標準としてまとめ上げる方法を示すアプリケーションプロファイル

本文書で提案する改善は、同様に、次のようにも規定される。

1. デポジットに対するレスポンスが新たな識別子タイプにより補強される。

2. デフォルトのマニフェスト文書が少なくとも 1 つの拡張として仕様に規定される。デフォルトのシリアル化が提供されるが、別のフォーマットをサポートする何らかのメカニズム（たとえば、コンテンツネゴシエーションなど）も提供される。仕様は、デフォルト以外のシリアル化が標準に対する独自の拡張を提供することを可能にする。

マニフェスト文書のシリアル化

デフォルトのマニフェスト文書は OAI-ORE [18] とその RDF シリアル化 [19] に基づくことが提案されている。OAI-ORE はウェブベースの複雑なデジタルオブジェクトの記述を可能にするために開発されたので、SWORD の理念に沿った方法でリポジトリオブジェクトの構造を記述するのに適している。実際、リポジトリの中には既に、ウェブページに RDFa を埋め込んだり、様々なフォーマットの ORE リソースマップを明示的に公開したりして、OAI-ORE を何らかの形でサポートする実装を行っているものもある。

OAI-ORE はウェブリソースの集合体を記述するものであり、上で述べたようなリポジトリ中のアイテムの状態を表現できるセマンティクスは含んでいない。OAI-ORE は（リンクトデータ [20] ムーブメントが依存しているフォーマットである）RDF で表現できるので、アイテムのワークフロー上での状態を表明する追加の RDF 表明文と OAI-ORE リソースマップの両者を含むマニフェスト文書を作成するのは容易である。RDF/XML でシリアル化した次のような基本的なマニフェスト文書の例を考えてみて欲しい。

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ore="http://www.openarchives.org/ore/terms/"
   xmlns:sword="http://swordapp.org/terms/">

  <!-- リソースを記述する ORE リソースマップを宣言する標準的な方法 -->
  <rdf:Description rdf:about="http://swordapp.org/rem/rdfxml/100">
    <ore:describes rdf:resource="http://swordapp.org/aggregation/100"/>
  </rdf:Description>

  <!-- リポジトリアイテムの記述 -->
  <rdf:Description rdf:about="http://swordapp.org/aggregation/100">
    <ore:isDescribedBy rdf:resource="http://swordapp.org/rem/rdfxml/100"/>
    <!-- リポジトリアイテムに関連するファイル -->
    <ore:aggregates rdf:resource="http://swordapp.org/object/100/file1.pdf"/>
    <ore:aggregates rdf:resource="http://swordapp.org/object/100/file1.pdf"/>
    <ore:aggregates rdf:resource="http://swordapp.org/object/100/file1.pdf"/>

    <!-- アイテムのリポジトリにおける状態に関する SWORD 関係 -->
    <sword:state rdf:resource="http://swordapp.org/terms/under-review"/>
```

```
    <sword:state-description
        rdf:resource="http://swordapp.org/repo/under-review"/>
  </rdf:Description>
</rdf:RDF>
```

これは、ore:aggregates 要素で示されているように 3 つのファイルを含むオブジェクトの記述を提供している簡単な ORE リソースマップである。ここではさらに、アイテムの現在の状態を表す SWORD の標準辞書用語（あるいはアイテムの現在の状態を表すリポジトリ固有の用語）を示す sword:state と、その状態のリポジトリにおける意味をリポジトリがクライアントに提供する sword:state-description という 2 つの関係の表明が続いている。

既に OAI-ORE を実装しているリポジトリもあるので、何も設定しなくても、リポジトリはおそらく上で示した以上の情報をリソースマップで提供できるだろうと考えられている。このような場合、SWORD 仕様は何の制限も加えるべきではなく、幅広く理解可能なマニフェスト文書を提供する OAI-ORE プロファイルとして機能すれば良い。したがって、仕様で規定するのは、リポジトリがクライアントに注意を喚起したいリポジトリアイテムの各ファイルについて ore:aggregates 要素が提供されなければならないというだけである。

その最終的な目的は、次の操作が可能になるように OAI-ORE の SWORD プロファイルが十分な情報を提供するのを保証することである。

- リポジトリアイテムに新規ファイルを追加する HTTP POST。たとえば、集合体 URI（上の例の http://swordapp.org/aggregation/100 ）に対して発行される。
- 既存のファイルを新規ファイルで置き換える HTTP PUT。 ore:aggregates 要素で参照されているリソースに対して発行される。
- 既存のファイルを削除する HTTP DELETE。これも ore:aggregates 要素で参照されているリソースに対して発行される。
- リポジトリアイテムの個々のファイルを取り出す HTTP GET。 ore:aggregates 要素で参照されているリソースに対して発行される。

これら 4 つの標準的な HTTP 操作は、リソースマップで記述されているように、リポジトリアイテムそのもの、あるいはデポジットされたパッケージに対するものではなく、オリジナルパッケージのデポジットで作成されたリポジトリアイテムのコンテンツに対するものであることに注意されたい。

状態記述

では、クライアントが sword:state-description 要素のリソースを逆参照した場合何になるべきなのか。SWORD の考え方に合わせると、これには、適切な情報を伝達するために特に atom:title 要素と atom:summary 要素を重視した Atom エントリ文書であることが推奨される。その利点は、Atom は拡張が可能であるので、この状態記述にさらに情報を追加することで恩恵を受けられることが明らかになった場合に、情報の追加が容易であることである。

しかし、これは Atom 標準からの逸脱であり、コミュニティからは代替案の提示や単純化が要求されている。その代替案の 1 つは、たとえば、単にこの情報を伝達する独自フォーマットを定義することである。

サンプル実装

本文書につながった作業の多くは、Symplectic 社 [21] において、SWORD 拡張をした AtomPub を使って数多くのデジタルリポジトリに完全な CRUD（作成、検索、更新、削除）API を実装する中で行われた[22]。

この作業の目的は同社の出版管理システム（Symplectic Elements [23] ）を様々なデジタルリポジトリに統合し、研究者が日常的に使用する出版システムにスムーズに統合されたリッチなデポジットクライアントを提供することであった。その機能は、コンテンツパッケージの使用が難しい個々のファイルを研究者がいつでもリポジトリにアップロードできるようにするものである。

この要件を満たすために、SWORD 拡張をした AtomPub の完全な実装が行われた。これは、すべてのリポジトリに一貫した API を提供するものであり、これによりリポジトリはクライアントに変更を意識させることなく自由に情報を交換することが可能になる。その結果は、リポジトリを意識する必要もないようにユーザを抽象化した、デジタルリポジトリに対するシンプルで効率的なインターフェースであった。このインターフェースは、「はじめに」で述べた SWORD の限界に答えるものである。

1. **リポジトリの作業の一部を外部システムが行わなければならない:** リポジトリをファイルコンテンツのオーソリティだと考え、ファイルはリポジトリに直接アップロードされる。
2. **アイテムがいつアーカイブできるかをユーザが知らなければならない:** シンプルなファイル管理インターフェースを提供しており、アイテムが厳密にいつ「完成した」かをユーザに明示的に尋ねることはない。また、変更は後でいつでも可能である。
3. **SWORD 用の AtomPub プロファイルが明確でない:** SWORD での使用に適した AtomPub のあらゆる機能を使用しており、その一部は本文書に活かされている。
4. **構造化パッケージに依存している:** 同様に、パッケージを使用しないことにより、特定のパッケージ形式をリポジトリが理解できなければならないという問題を回避している。

SWORD 4 プロジェクトの期間中に、Symplectic 社は同社のリポジトリツールシステムに基づいた、SWORD 次期バージョン案を実装したオープンソースのサーバシステムをコミュニティに提供する予定である。これにはコミュニティからの意見を求めるために、SWORD 標準の方向性とその実装に関する数多くの追加の技術提案を含め、プロジェクトが SWORD 標準の次期バージョンを完全に定義するのを助ける予定である。これは SWORD 2.0 標準の最終的な実装になるわけではなく、開発者が検討を行うためのサンプル実装である。このサンプル実装で使用されるリポジトリプラットフォームはコミュニティの要求に基づいて選ばれる予定である。

結論

SWORD の次期バージョンの目的はユーザにリッチで魅力的な経験を与えるために必要なツールをクライアントに提供することである。そのために、リポジトリがコンテンツをどのように処理したかをクライアントがより確実に理解できる機能を導入する。また、明確に定義された識別子を採用することにより処理の追跡を可能にし、リポジトリの双方向性を高める。

本文書では、SWORD 4 プロジェクトが行う予定である 2 つの主要な開発について述べた。

1. 現行のデポジット受領通知に使用されている識別子に厳密なセマンティクスを与え、さらに、リポジトリにとって有用な識別子を2つ追加する。

2. リポジトリアイテムの内容を記述でき、状態についての基本的な情報を提供できるデフォルトのマニフェスト文書のシリアル化を規定する。

本文書は、SWORD標準にこれら2点を追加することにより、この標準がデポジットのライフサイクル全般を管理する方向に導かれることを示したが、冒頭で述べた限界のすべてに答えたわけではない。現段階では、デポジット以外の操作に対する認証と承認の方法、デポジット後にパッケージの一部を変更する方法、パッケージ化を行うことなく単一のファイルのみをデポジットする方法、デポジットに使用されたパッケージ化フォーマットをより正確に記述または規定する方法などが明らかに欠けている。これらすべてはSWORDが将来のバージョンで取り扱うべき課題である。

本文書で示した開発の詳細やサンプル実装は次期バージョンに取り組むための提案として示したものであり、決して最終的なものではない。むしろ、これらはリポジトリコミュニティで既に使用されていたり、情報システムの生態系の中でリポジトリがその役割を果たすために必要だと考えられている、明らかなベストプラクティスから抽出したものである。本文書の目的はこれらの提案をコミュニティに行い、その改善や変更、限界について意見を求めることである。オープンソースのリファレンス実装を開発しコミュニティによる全般的評価を受けることにより、これらの提案がどのような結果になり、デポジットの状態管理の改善にどのように使用できるかを知ることも可能になると思われる。

参考資料

[1] SWORD: Home page: http://www.swordapp.org/ ; Specification: http://www.swordapp.org/sword/specifications
[2] JISC: http://www.jisc.ac.uk/
[3] DSpace: http://www.dspace.org/
[4] EPrints: http://www.eprints.org/
[5] Fedora: http://www.fedora-commons.org/
[6] arXiv.org: http://arxiv.org/
[7] Microsoft Zentity: http://research.microsoft.com/en-us/projects/zentity/
[8] Intrallect IntraLibrary http://www.intrallect.com
[9] Facebook SWORD App by Stuart Lewis: http://fb.swordapp.org/
[10] Microsoft Word SWORD Add-in. Presentation from OR10: http://research.microsoft.com/en-us/um/redmond/projects/authoring/090518-pablofe-connecting%20authors%20and%20repositories_%20through%20sword.pptx ; Video Tutorial: http://www.youtube.com/watch?v=2_M2gfUyVzU
[11] Open Journal System, SWORD plugin: http://pkp.sfu.ca/wiki/index.php/Ojs_plugins#SWORD_1.2_Repository_Deposit
[12] The Atom Publishing Protocol: http://bitworking.org/projects/atom/rfc5023.html
[13] Metadata Encoding and Transmission Standard (METS): http://www.loc.gov/standards/mets/
[14] Atom: http://www.ietf.org/rfc/rfc4287.txt
[15] Atom Multipart Draft: http://tools.ietf.org/html/draft-gregorio-atompub-multipart-04

[16] Guidelines for Writing an IANA Considerations Section:
http://www.faqs.org/rfcs/rfc2434.html
[17] WebDav: http://www.webdav.org/specs/rfc2518.html
[18] Open Archives Initiative - Object Reuse and Exchange (OAI-ORE):
http://www.openarchives.org/ore/
[19] OAI-ORE RDF/XML serialisation:
http://www.openarchives.org/ore/1.0/rdfxml.html [日本語訳]
[20] Linked Data: http://linkeddata.org/
[21] Symplectic Ltd: http://www.symplectic.co.uk/
[22] Symplectic Repository Tools:
http://www.symplectic.co.uk/products/repository-tools.html
[23] Symplectic Elements publications management system:
http://www.symplectic.co.uk/products/publications.html

連絡先

メーリングリスト: https://lists.sourceforge.net/lists/listinfo/sword-app-tech

プロジェクトマネージャー: Adrian Stevenson, UKOLN, a.stevenson@ukoln.ac.uk

著者: Richard Jones, Symplectic Ltd, richard@symplectic.co.uk

連絡先: http://www.swordapp.org/contactus