Life Cycle General Use Case

SIMILE General Use Case #2: Lifecycle Management of Digital Objects

Modelled after Section 2 of ***SIMILE Use Cases Document***

Key Challenges being addressed by this use case

A core function of an OAIS-compliant archive is to maintain sets of submissions and distribution agreements that define the SIP and DIP formats that the archive will support, and with knowledge of these agreements, design appropriate AIP structures, ingest and maintenance processes such that these agreements may be successfully fullfilled over time.

Capabilities that we believe to be important in accomplishing this objective include:

- policy expression, management & enforcement
- digital object modelling
- interaction with a digital object via an API
- definition, insertion, and management of views on objects
- event modelling
- modelling and connecting triggers - responses to events
- audit metadata generation, and audit services

Core of the generic problem: Communities coming to table, seeking agreement with the Library about what submission agreements they will have with the Library. The library must have some way to codify that SIPs submitted get turned into AIPs. And, over time, how AIPs get acted upon over time for the archive to success in its mission. Including, on the outbound side, what DIPs we have committed to (or wish to commit to).

Further, throughout this process, all of these changes, omissions, extension have been made; must be able to answer the question, *what happened* over the course of our stewarfship of this AIP. Answer not only for ourselves, but for third parties; nobody needs to "trust us," but rather there is a way to test/validate that the archive is being run according to agreed-upon/committed-to policies.

1. Conversation occurs between Library and Community about what SIPs/DIPS are desirable
    - and what kinds of services are implied, along path from SIP through AIP to DIP
        - e,g. transformation, indexing, etc.
    - submission agreement capture
        - these are the (kind of) services we expect
        - codification of this agreement
2. AIP Design: Bearing in mind the community-specific nature of the SIP and DIP Agreements, as well as the Library's *overall* committment to preservation and stewardship over time, *someone* in the Library make decisions about the required structure and information content of **AIPs** for content submitted by this community.

    - some of these decisions are primarily content- or format-specific, and slice across communities and collections; for example, reflecting Library policy for preserving documents in PDF format or images.
    - some of these decisions reflect the nature of the specific agreements that the Library has made with this community; e.g. indexing or viewer support, or particular community-specific schema.
3. SIP/AIP Transformation Design
    - *and* AIP/DIP Transformation Design
    - defining what the process actually looks like (how it is actually carried out)
        - the transformation(s) required
        - the services required to bring these about
    - someone must define the rules and sub-workflows that must be implemented
4. Preservation Planning
    - How should AIPs be transformed and/or augmented over time in response to preservation policies defined by the Library
    - The Library agrees with the Community to some presservation level, which is defined by a sett of policies which they have published.
5. Event definition, selection, prioritization
    - Collection management is fundamentally based on binding services to events
    - The Library defines those events that are important for it to be able to specify the kinds of transformation steps that are required
        - as well as the kinds of preservations "triggers" that it finds important
6. View generation and Maintenance
    - A DIP is a view on an object with a committment from the Library to support
    - Over time, the Library, in response to shifting demands from the COmmunity may introduce *additional* DIPs (aka *views on AIPs*) and commit to supporting these views
    - The Library must be equipped to answer the questions:
        - what is the complete set of views we are committed to
        - what is the complete set of services required to support these committed-to views
7. Audit/Provenance Metadata Generation & Use
    - *How did this object get to be the way that it is?*
    - Libraries and/or Third Parties have a need to be able to answer the question:
        - *which transformations have occured for this object?*
        - In response to which rules, policies, etc.
    - Third parties may have an *additional need* to assess over time the consistency of the operation of the archive with the Library's stated and committed-to policies
        - *Have objects actually been transformed as agreed-to*
        - *Are the overall service levels of the archive being met?*

Prototype/Demonstrators: Lifecycle Management of Digital Objects

1. **Modelling and Management of SIP & DIP Agreements**
    - Structure (of model)

        - design approach
    - "Negotiation" workflow
        - sequence of managed steps leading to mutally-"blessed" agreement persisted in repository
    - Presentation (of instance)
        - authoring/creation
        - viewing
        - includes pre- and post-signature agreements
    - Binding of "agreement" objects to "policy" objects
        - association of "clauses" within natural language agreements with policy specifications
    - Binding of "agreement" objects to "service" objects
    - Agreement Review Interface: Show me agreements with *xxx, yyy, zzz*/ characteristics
    - **Deliverable:** Stand-along Agreement Manager demonstrating the display and management of SIP & DIP agreements, esp. binding of agreement elements to functional policy objects that implement agreement elements.
2. **Repo-based AIP demonstration**
    - Create model AIP
    - Implement on M-Fedora
    - Sketch **DIP -> AIP** and **AIP -> SIP** transformaton models
    - **Deliverable:** Demonstration of exemplar AIP implemented on M-Fedora. Demonstration of DIP disseminator(s) providing "views" on this AIP. DEmosntration of **SIP -> AIP** transformation capability.
3. **Joseki-based DIP demonstration**
    - one approach to "views"
    - **Deliverable:** Demonstration of Joseki service layer bolted on Fedora or Dspace instance. Implementation of Joseki queries that generate specific exemplar DIPs.
4. **Fedora-based DIP demonstration**
    - another approach to views
    - **Deliverable:** Demonstration of dissemination of exemplar DIP from Fedora.
5. **Policy-based Transformation Engine**
    - Workflow-oriented policy instance
        - *For this collection and this type of submission, collect XXX metadata and do YYY transformation*
    - Policy engine
    - SIP submission validation
        - Template-based metadata collection, type validation
    - Policy-driven transformation
        - remote, message-based service invocation (e.g. SOAP-based indexing submission, image transformation, whatever)
    - **Deliverable:** Demonstration of workflow-oriented policy interpretor. Demonstration of ability to invoke transformation-oriented (e.g. JPG -> TIF transformation, PDF -> TXT extraction) or indexing service calls due to triggering of policy.
6. **Event-driven Preservation Policy Engine**
    - Management-oriented policy instance
    - Policy engine
    - Management-oriented, event-driven service invocation

        - Calendar-based integrity check
        - Calendar-based transformation
        - Alert-based transformation (e.g. due to schema change)
        - (related) Instance changed-based service invocation
    - **Deliverable:** Demonstration of preservation-oriented policy interpretor. Demonstration of ability to invoke appropriate preservation-oriented service calls due to triggering of policy. Possible approach: Use existing Ponder implementation (Imperial).
7. **Event modelling**
    - Event base structure
        - Specialization to specific event
    - Relationship to `Harmony ABC Event Model`?
        - In ABC model, events have *inputs, outputs, acts* and *contexts*
        - specific resource attributes will depend on the type of resource and the domain-specific requirements
    - Propagation of event
    - Referencing of event to History "view"
        - e.g. named Joseki query to
    - **Deliverable:** Demonstrate event specification and "watch" capability in selected repository testbed (e.g. Fedora or Dspace). Demonstrate management of events as first-class objects within repo, including persistence, management and dissemination (esp. views on event metadata), as well as management of the event object model itself.
8. **Event-driven Audit Demonstration**
    - Audit-oriented policy instance
    - DO action-based event as policy "trigger"
    - Audit record model
        - incl. event object reference or instance
    - **Deliverable:** Demonstration of audit-oriented policy interpretor, including expression of "event-watching" policies. Demonstration of ability to invoke appropriate audit-oriented service calls (e.g. logging of specific transformations, changes, distruptions) due to triggering of policies.