I Introduction

MOSAIC is a cloud portal for hosting scientific, academic, and technical publications. MOSAIC bridges the gap between traditional academic publishing and the new world of software- and data-driven research platforms, which feature interactive reading experiences and open-access data sharing.

Today there are many online services which can be called AST (Academic/Scientific/Technical) Portals. These resources, catering to an academic and technical audience, aggregate research articles, books, journals, and publicly accessible research data. AST Portals typically link to web pages devoted to individual publications, with each page displaying citations, abstracts, keywords/topical classification, and (when appropriate) links to download documents in e-reader (typically PDF) formats. Some AST Portals are curated by individual publishers; others (such as Sciverse, Mendeley Data, and Dryad) are scientific projects incorporating content from many publishers.

Because they were designed for traditional publications, current AST Portals have limited support for technologically sophisticated features of modern publications: annotating document text with linked data, and joining publications with open-access data and interactive presentations. Publishers have therefore sponsored several projects and standards, including the FAIR (Findable, Accessible, Interoperable, Reusable) initiative, FORCE11 (Future of Research Communication and e-Scholarship), and the Research Object Protocol. These initiatives are designed to improve research portals' support for cutting-edge publication technology. In the words of the FORCE11 "Manifesto":

> While not disputing the expressive power of the written word to communicate complex ideas, our foundational assumption is that scholarly communication by means of semantically-enhanced media-rich digital publishing is likely to have a greater impact than communication in traditional print media or electronic facsimiles of printed works. However, to date, online [publications] have tended to replicate print forms, rather than exploit the additional functionalities afforded by the digital terrain. We believe that digital publishing of enhanced papers will enable more effective scholarly communication, which will also broaden to include, for example, better links to data, the publication of software tools, mathematical models, protocols and workflows, and research communication by means of social media channels. https://www.force11.org/about/manifesto

Unfortunately, publishers have not significantly adopted or embraced proposals such as the Research Object Protocol. As a result, authors and researchers themselves have fewer incentives to embrace these paradigms, despite the efforts of groups like FORCE11 to promote them. While scientists increasingly publish raw data — part of an emerging paradigm prioritizing replication and transparency — very few data sets are published as conformant Research Objects. Even when data sets do have this extra structure, the research portals which link to their associated publications have no tehnological means to document or provide access to the enhanced features afforded by the Research Object Protocol.

This explains why a new Academic, Scientific, and Technical publishing platform is needed. MOSAIC steps into this void first by instantiating a new AST Portal and second by providing the Cloud-Native technology needed to reproduce this portal in other conexts, so that publishers and other institutions can create their own versions of MOSAIC or reuse its components in other projects.

In addition to MOSAIC itself, the technology driving the platform can be deployed as a suite of three products that can be licenced individually or collectively — the MOSAIC SDK, NDP CLOUD (Native-Driven Platform for the Cloud), and VersatileUX (a front-end development toolkit).

This summary and accompanying slides will describe the current technological limitations that MOSAIC addresses, outline MOSAIC as a whole, and then review the MOSAIC SDK, NDP CLOUD, and VersatileUX products.

1.1 *How the Research Object Protocol is Supposed to Work*

Unlike a data set which just exposes raw data, a Research Object (RO) combines data and code into a bundle that promotes interactive exploration and reuse, adding value to associated publications. To fully leverage this added value, the Research Object Protocol was designed to ensure that RO publications can seamlessly interoperate with research portals and publishers' web sites.

According to the standard, when an author publishers a RO — either on a specialized academic site like Mendeley Data or a generic service like Github — authors would notify the maintainers of AST Portals (those which index the corresponding publication) that a new data set, associated with the corresponding publication, is now publicly available. Because ROs have a fixed structure, portals' software can then, in principle, automatically extract metadata from the RO and use this metadata to update the relevant web pages.

These web pages can then, first, include a link for readers to download the data set itself as a zipped folder, and, second, present basic information about the data set: its software requirements, file formats, download/unzipped size, programming language(s) used, update and version info, and instructions on compiling/using the RO code. Third, web applications (and other kinds of software) can then use the RO metadata to display an overview of the RO code — for instance, a summary of important data types and important procedures implemented on them. Technical overviews of RO code serves as pedagogical overviews of research methods: the data structures and algorithms used to compute research findings encapsulate the data collection instruments (which can be lab equiment but also surveys, corpora, and other social-science data sources) and the experimental protocols which have informed the published work.

In short, ROs are a logical extension of a published book or article. By sharing code alongside raw data, each RO includes a code base whose design captures the essential structural and methodological features of the research. For example, by modeling the information entered into the system by data collection instruments, the RO code base clarifies the system's scientific theory and epistemological paradigms. It distinguishes input data points from calculated data values, and demonstrates the requirements and presuppotions concerning input data and how derived values are computed. The RO code thereby bcomes a concrete prototype of the experimental and analytical methodology described theoretically in the publication.

In principle, then, AST Portals can use Research Objects to build summary presentations of published research. Like an abstract, these summaries can show readers a concise overview of a publication to help them decide whether they will benefit from rading and/or downloading the publication and/or data set as a whole.

1.2 *But the Research Object Protocol has not Been Implemented!*

Unfortunately, the RO features just described have not yet found their way into existing AST Portals. Current technology does not interoprate with Research Objects on either the simpler metadata level or the more sophisticated architectural level. This gap is a combination of three distinct problems:

1. First, there is no standardized way for authors to notify publishers about new Research Objects. Some data set portals, like Mendeley, have web forms where researchers can enter information about their ROs. According to Elsevier, the metadata in Mendeley is then internally shared with other Elsevier portals, such as SciVerse. However, it is not clear when new technology which is supposed to integrate Mendeley Data and SciVerse will actually come online. Other publishers, meanwhile, do not appear to have any mechanism at all for interoperating between AST Portals (which aggregate publications) and data set portals (which can aggregate Research Objects).

2. Even if authors *can* refer AST Portals to published ROs, the AST Portals need sufficient capabilities to access and interpret RO metadata. This requires portal software that can read such metadata, and also that web templates include sections for displaying ROs' technical information (like programming language and file format), the kind of basic information which can be automatically extracted according to the Research Object Protocol.

3. Moreover, to fully leverage the value of Research Objects, AST Portals need to interpret RO metadata at a higher scientific level. This becomes possible when RO metadata presents summary information about the experimental protocols and computational methods employed via RO code. However, although the Research Object Protocol recommends that data bundles include this sort of higher-level metadata, the protocol does not specify how such scientific information should be represented and how it should be used by protocol-compliant software.

   According to the Research Object bundle specification:

   > The specification for the RO model does not mandate any particular form for the representation of Research Objects. ... Annotations about this research object and its resources ... may be present, [and] provides additional metadata or descriptions which are somewhat about or related to the research object or some of its aggregated resources.
   > https://researchobject.github.io/specifications/bundle

Extending the underlying RO model, developers have specified more detailed models such as "Wf4Ever":

> The focus of Wf4Ever is on workflow preservation (and more specifically workflows supporting scientific investigation). Thus the objects that will be described using the model are inherently workflow-centric. Although the basic infrastructure (aggregation + annotation + domain vocabularies) that the RO model supports is applicable to many situations ... the specific vocabularies defined within the Wf4Ever RO model are intended to support those Research Objects that have workflows (methods) as their primary content. https://wf4ever.github.io/ro/2016-01-28/

In short, a well-established model is provided for workflow-centric Research Objects, but comparable specifications have not emerged for other kinds of technical projects. In the general case, publishers have to define their own RO models. That is, they have to specify what sort of metadata ROs should expose so that information about ROs can be integrated into publishers' overall online resources.

Because publishers have been unwilling to develop their own Research Object technology, authors do not have clear guidelines about how to document and annotate ROs even once they embrace the Research Object paradigm. This has apparently caused an impasse wherein neither authors nor publishers model sufficiently detailed requirements for one another, to complete the Research Object circuit. In particular, scientists do not have RO data models for most academic disciplines which are as fine-grained as the Wf4Ever Ontology. While this situation surely reflects technologically inertia, it also testifies to the conceptual gap between Semantic Web Ontologies and scientific data models, a gap which has been noted by many scientists themselves.

Indeed, scientists have criticized contemporary data sharing standards — e.g., the Semantic Web — for lacking conceptual precision and failing to capture the essence of scientific procedures. In the words of Martin Raubal and Benjamin Adams,

> the Semantic Web is still not semantic in the human sense because it does not sufficiently account for people's cognition, i.e., human conceptual representations and reasoning... [K]nowledge representations underpinning the Semantic Web should afford two important human cognitive tasks: the efficient calculation of semantic similarity ... and combinations of concepts.... However, the existing logical foundations of the Semantic Web — description logics and rules — presume a set-based classification scheme that does a poor job of facilitating these operations.
> http://www.semantic-web-journal.net/sites/default/files/swj37_0.pdf

Raubal and Adams have developed their own system for representing scientific data, called Conceptual Space Markup Language (CSML), which is equal or superior to RDF Ontologies for describing scientific metadata. However, although the Research Object Protocol does not specifically require RDF (Resource Description Format), its tooling and specifications clearly show the influence of the Semantic Web. This has prevented alternative models like Conceptual Spaces — originally proposd by the Cognitive Scientist, Peter Gärdenfors — which are intended to be more accurate models of scientific data, based on investigation of both human conceptualization and scientific practice.

In contrast to the existing Research Object model, MOSAIC — an acronym for *Multiparadigm Ontologies for Scientific, Academic, and Technical Publications* — provides a detailed structure for describing scientific data. MOSAIC unifies diverse models and theories of scientific metadata, such as Conceptual Space Markup Language, into a canonical but flexible format that ensures interoperability between RO code and AST Portals' software. MOSAIC, furthermore, models scientific metadata with rigorous attention to scientific practice (and research methodology in general, across academic disciplines). In short, for MOSAIC, metadata is not just a technological artifact but a conceptual summary of scientific theories, practices, and conceptualizations.

II MOSAIC in Detail

2.1 *The MOSAIC Architecture and Data Models*

A MOSAIC portal combines a central web service with a collection of independent Research Objects. Not every document indexed by MOSAIC must have an associated Research Object, but the technological design of MOSAIC prioritizes integrating research portals with downloadable Research Objects.

In particular, MOSAIC encourages the creation of *self-contained* Research Objects. MOSAIC assumes that the typical Research Object indexed on a MOSAIC portal will be a downloadable code-and-data bundle with *minimal external dependencies*. It should be easy for readers to get started exploring and interacting with Research Objects without complicated downloads or installation of additional software. Authors are certainly free to *enhance* Research Objects with content tailored to specialized software — for example, sophisticated data visualization features, or integration with software commonly used in the relevant research fields. However, these extra capabilities augment the core of the RO, which should provide a standalone mechanism for researchers to understand, visualize, and reuse the bundled code and data, without requiring external software or code libraries.

In order to help authors create self-contained Research Objects, MOSAIC defines a *Reference Application Kernel* (*RAK*) which presents a canonical format for bundling research data and code into a self-contained package. Each RO bundle using the RAK format provides a standalone, desktop-style Dataset Application that offers an entry point to the raw data. By design, it should be easy to download the RAK bundle from MOSAIC and then compile and launch the Dataset Application, so readers can begin exploring the data set with minimal hassle.

The RAK format includes other features as well, such as code export and testing, which provide a more detailed access to the published data (including several command-line executables built alongside the Dataset Application). However, the Dataset Application presents a useful introduction to the underlying data set, which helps users get oriented to the data set in its broad overview before they start to examine it in finer detail.

Technically, MOSAIC RAK bundles are code repositories based on the Qt platform for C++ Applications. In RAK, each data set features C++ code which has no dependencies apart from Qt itself. The Dataset Application should compile and run automatically from inside Qt Creator. Because Qt is an advanced desktop GUI development platform, it allows ROs to present compelling, interactive graphics and GUI components — including 2D and 3D visualizations, tables and charts, context menus, dialog boxes, explanations of technical terms, embedded PDF viewers, and other features which make Dataset Applications an interactive and pedagogically useful tool for exploring raw data, customized for each data set.

Research Objects indexed by MOSAIC do not have to use the RAK model. However, RAK provides a Reference Implementation demonstrating how RO metadata can be described for it to be automatically processed by a MOSAIC portal. Developers using different RO models can examine how RAK bundles expose data to MOSAIC so as to create analogous metadata for their own bundles.

2.2 *The MOSAIC SDK and Data Modeling Features*

To facilitate implementation of RAK bundles, MOSAIC provides several development tools and code libraries which can help set up a development environment tailored to the MOSAIC ecosystem. These assets include the following:

Build Tools Since it is designed to run inside Qt Creator, the RAK build systm is based on *qmake*, which is Qt's layer atop the C/C++ "make" system. MOSAIC's build system extends *qmake* to simplify the integration of RAK applications with MOSAIC portals and with other scientific software that may be available as enhancements for RAK applications, depending on their discipline and topics.

Hypergraph Code Libraries and Parsers MOSAIC defines a multiparadigm representation for scientific data based on Directed Hypergraphs. This representation is flexible enough to accommodate many different modeling paradigms — including conventional OWL Ontologies, CSML-style Conceptual Spaces, and several other workflow and linguistics-based models — while at the same time conducive to self-contained Dataset Applications. The MOSAIC SDK provides code for creating hypergraphs from text documents and integrating hypergraph data into C++ applications.

Scripting and Testing The MOSAIC SDK includes components that developers can use to create test suites for Research Objects and to design a customized scripting environment. Selected functions from the RO code are exposed to a Runtime Reflection engine, which allows these functions to be invoked by scripts, including scripts composed for unit and integration testing. Functional annotations provide detailed Requirements Engineering for application procedures, including scientific details like statistical scale (Nominal/Ordinal/Interval/Ratio), dimensions, ranges, and declarations of side effects.

MOSAIC Interop Applications built with the MOSAIC SDK can also automatically interoperate with MOSAIC portals. The SDK allows Dataset Applications to be equipped with special features for different user roles, including authors and editors as well as ordinary readers. Authorized users can automatically notify MOSAIC, from within their Dataset Application itself, about a new or updated version of their Research Object.

Architecturally, MOSAIC is developed as a Cloud/Native Hybrid featuring a collection of independent native applications (the RAK Research Objects) connected to a central cloud service (the MOSAIC portal). The portal itself is also a Cloud/Native Hybrid, in that a miniature version of a MOSAIC portal can run as a standalone desktop application. MOSAIC is implemented to allow seamless integration between native and cloud components — for example, using non-GUI Qt libraries as the core of the server-side code — so as to minimize data marshaling when the cloud service communicates with native endpoints.

As described above, not every Research Object indexed by a MOSAIC portal will use the MOSAIC SDK or MOSAIC's RAK format. Research Object developers are free to adopt their preferred RO models and communicate with MOSAIC via an API. However, the RAK model can still be used in this case as a reference for other kinds of Research Objects, to clarify the requirements and proper usage of the MOSAIC API.

2.3 *MOSAIC Products*

The MOSAIC SDK, as just described, enables developers to create ROs that seamlessly interoperate with MOSAIC portals. Because of MOSAIC's unique design requirements — supporting a rich, multiparadigm semantics for scientific data and centering a network of self-contained desktop applications — the MOSAIC SDK introduces innovative features that companies may find useful outside the context of academic publishing/Research Objects. The MOSAIC system prioritizes Modular Native Design, for applications that are flexible and versatile from a user's point of view, while also securing a software-development methodology that facilitates transparent Requirements Engineering, rigorous code verification, and demonastrable compliance with legal and operational standards throughout the lifetime of a project.

In addition to the SDK, MOSAIC utilizes technologies targeted to both server-side (Cloud) and client-side (Desktop) components. On the server side, NDP CLOUD is a framework for deploying cloud services tightly integrated with native applications. NDP CLOUD applications can be tested and developed as standalone desktop applications before being uploaded to cloud servers. NDP CLOUD also eliminates most of the technological gap between web and desktop implementations. For the most part, NDP CLOUD uses the same code libraries and programming paradigms as desktop applications (such as Qt/C++), so that client-server interop becomes analogous to networking between two desktop applications. For this reason, NDP CLOUD can be a good solution for developers who want to use cloud services to augment the capabilities of desktop software, enabling peer-to-peer messaging and seamless cloud backup and personalization to enhance the desktop experience.

On the client side, VersatileUX is an GUI application framework which leverages the peer-to-peer and personalization capabilities of NDP CLOUD. VersatileUX supports a highly modular approach to application development, with tight correlation between data structures and the GUI elements used to display them. Any VersatileUX software is a composite of semi-autonomous modules, each of which is responsible for reading or receiving data conforming to specific structures (from a file or a network) and presenting these data structures in specialized GUIs. GUI and data-structure components are bound together via strong typing. Because VersatileUX components are developed in isolation, testing and compliance verification is simplified for the application as a whole. At the same time, each VersatileUX module can design its own scripting and personalization features, so the overall application can be granularly fine-tuned to the needs of each user.

NDP CLOUD and VersatileUX work together to promote personalization and interoperability. Individual desktop applications, built with the aid of VersatileUX, can connect to an NDP CLOUD service so as to track user identity across application instances (or even across different applications), while working through the cloud service for data backup and to connect multiple users, for purposes of messaging and collaboration. VersatileUX modules can be fine-tuned via user preferences and application-specific data obtained from NDP CLOUD services, updating application state via Runtime Reflection. Reflecting MOSAIC's emphasis on complex semantics for scientific data, NDP CLOUD and VersatileUX interoperate via sophisticated protocols for exchanging strongly-typed data structures between clients and servers and between clients themselves (via cloud messaging). The key difference between this Cloud/Native Hybrid architecture and conventional client/server architecture (including other Cloud/Native paradigms) is that both server- and client-side code uses strong typing, and a single type system is sustained across all client and server components.

Companies can license the MOSAIC SDK, VersatileUX, and/or NDP CLOUD whether or not they host publications on MOSAIC or host their own instance of a MOSAIC portal.

2.4 *For More information*

Please see the accompanying slides or request a meeting or phone conference to discuss MOSAIC in greater detail!