WHITE PAPER

The MOSAIC Publication Repository: Supplemental Archives, Structured Research Reporting, and Plugins

Overview

In recent years — as publishing has become more "digitized" — there have emerged many online platforms for scientists to share and publish their research work, including both raw data and scientific papers. These portals generally provide an index/search feature where readers can locate research based on the name of the author, title of the publication, keywords, or subject matter, along with a document viewer where readers can view the abstract of each publication (and sometimes full text of the article) and other publication details (such as co-authors, date of publication, bibliographic references, etc.). However, these portals offer only limited features for interactively exploring publications, particularly when it comes to examining data sets and/or browsing multimedia assets associated with publications, such as audio, video, interactive diagrams, or 3D graphics.

The "MOSAIC" system (for "Multi-Paradigm Ontologies for Scientific and Technical Publications") is a suite of code libraries developed by LTS allowing institutions to host publication repositories free of the limitations of existing scientific document portals. With MOSAIC, each publication can be linked to a supplemental archive which contains information about the author's research methods, data sets, and focal topics. If desired, these archives can include machine-readable representations of full publication text, to support advanced text-mining techniques across the repository.

Using MOSAIC, developers can implement a hosting platform for all of this supplemental material, in addition to publications themselves, where the platform provides software enabling readers to browse and access supplemental archives and their data sets and/or methodological descriptions. A MOSAIC publication repository, also called a MOSAIC "portal," is a structured collection of data and documents which can be hosted via web servers (including fully encapsulated and containerizable cloud services) implemented with the help of MOSAIC libraries.

MOSAIC repositories can serve as general-purpose portals, hosting academic papers covering a broad range of topics. Alternatively, MOSAIC repositories can serve as targeted portals hosting papers focused on more narrowly focused subject domains. Organizations can utilize MOSAIC internally as the basis of a private document management system (DMS), or to provide (public) open access to complete publications, including human-readable and machine-readable full text and all supplemental content, with no restrictions[1]. MOSAIC can be customized for different subject areas by incorporating domain-specific ontologies into document-search features as well as machine-readable document-text annotations.

MOSAIC is designed so that the software to access publications and publications' supplemental archives can be embedded in scientific-computing applications via MOSAIC *plugins*. This ensures that publications and data sets can be examined interactively with the same software that scientists employ to conduct research. MOSAIC also introduces a *structured reporting system* (MOSAIC-SR) for describing research/experimental/lab methods/protocols. Supplemental archives, plugins, and the structured reporting system are outlined below.

---

[1] Excepting sensitive/private information which might be provided via a dedicated component within the portal with specific features for authors, editors, and administrators. Any project which *does* restrict access to some part of any document requires a commercial MOSAIC license

Part I: MOSAIC Components

Supplemental Archives

MOSAIC supplemental archives are additional resources paired with MOSAIC publications. In general, supplemental archives may include raw data, descriptions of research methods, or annotated data linked to publication texts. Each supplemental archive could have any of the following resources:

1. Interactive versions of the publication, with annotations indicating important concepts and phrases, perhaps aggregated into a "glossary" defining technical terms;
2. Machine-readable representations of document texts, with special-purpose character encodings designed to facilitate Text and Data Mining (TDM);
3. Structured files containing raw data discussed in the publication, along with interactive software allowing scientists to access and reuse the data;
4. Detailed reports of the author's research methods and experimental setup and/or protocols, conforming to the relevant standards with respect to the publication's subject classification — for instance, the "Minimum Information for Biological and Biomedical Investigations" (MIBBI) includes about 40 specific standards for different branches of biology and medicine;
5. Representations of analytic methods and algorithms underlying the research findings, which are provided directly via computer code or indirectly via formal descriptions of computational workflows;
6. Self-contained computer software which demonstrates code that the author developed and/or used for analyzing/curating research data;
7. Multi-media assets such as audio or video files, annotated images, 3D graphics, interactive 2D/3D plots and diagrams, or other kinds of non-textual content which needs to be viewed with special multi-media software.

The contents of a supplemental archive will be different for different publications, depending on whether the archive contains specific raw data or just a summary of the methods used to obtain the raw data. In the former case, a typical archive will include a *data-set application*, or a semi-autonomous software component allowing researchers to study and visualize the data set, typically provided in turn via data files that are also located in the archive. The data-set application will, in general, provide both a visual interface for raw data and code libraries for computationally manipulating this data; typically raw data files will encode serialized data structures that can be deserialized by code (e.g., C++ classes) included in the data-set application. MOSAIC includes generic implementations of a data-set explorer ("MdsX"), which can be adopted to the specific kind of information that needs to be serialized for a given publication. As a result, data-set application can then be packaged as a plugin to existing scientific software within the relevant scientific discipline.

MOSAIC Plugins

By design, MOSAIC Portals will provide a suite of plugins for existing scientific software, allowing publications and supplemental archives hosted on the portal to be read and examined within computer software associated with each publication's subject matter. For example, articles about chemistry could be read within IQMOL, a molecular visualization program; articles about cellular biology and bioimaging could be read within CAPTK (the Cancer Imaging and Phenomics Toolkit), an image-processing application; articles discussing novel computer code could be read within QT Creator, an Integrated Development Environment (IDE) for programming; etc. The advantage of accessing a publication and a supplemental archive within actual scientific software is that it allows research work to be understood, evaluated, and reused within the computing environments which scientists typically use to conduct professional research. This is different than existing science publication portals, which generally rely on web browsers to access supplemental materials like data sets and multimedia files — in this standard setup the software ecosystem wherein readers examine published research is fundamentally separate from the software platforms where research

is actually conducted. This example demonstrates how existing portals are *limited* in their ability to share research in rigorously reusable and replicable ways — and how MOSAIC offers a potential improvement.

Embedding presentations of their research within existing scientific software has the added benefit for authors of making their work more practically and immediately useful for the scientific community. Such presentations establish the computational framework for pragmatically deploying their techniques in real-world scientific contexts, accelerating the pace at which research work can be translated to concrete scientific (and clinical/lab/experimental) practice. As one example, research involving novel image analysis techniques could be packaged so as to target a MOSAIC plugin for bioimaging software such as CaPTk, so that readers could actually run the author's code as a CaPTk module.[2] This is important because such functional assessment and adoption of novel contributions is harder to carry out if a body of research work is described indirectly within article text, as opposed to being concretely implemented within a specific scientific application.

Another benefit of using plugins to access supplemental archives is that the host application will usually provide more sophisticated multimedia and data visualization capabilities, compared to static PDF images or even interactive web portals. Publishers have begun to develop online platforms for browsing research papers in conjunction with multimedia content such as interactive diagrams and 3D graphics — a physical model of a protein or a chemical compound, for instance, can be viewed online via WebGL; such graphics could even be embedded as an "**iframe**" within an HTML version of the publications. Publishers consider this to be cutting-edge technology. However, the same molecular 3D model, when viewed in IQmol, can be enhanced with many additional visual features, representing bonds, orbitals, torsion angles, etc. The multimedia experience of exploring chemistry data in custom software like IQmol is therefore much greater than the experience of generic web multimedia, which means that the scientific software is a better forum for showcasing novel research.

MOSAIC Structured Reporting (MOSAIC-SR)

The MOSAIC structured reporting framework (MOSAIC-SR) includes tools to help authors develop interactive presentations supplementing academic documents, and specifically to use supplemental archives to document how their research has been conducted. With MOSAIC-SR, authors can implement or reuse code libraries that report on research/experiment methods, workflows, and protocols. The MOSAIC-SR information may be structured as a "minimum information checklist" conformant to standards such as those collectively gathered into the MIBBI recommendations; in this case MOSAIC-SR would be applied by implementing object models instantiating MIBBI policies. Alternatively, MOSAIC-SR reports can be derived from actual computer programs simulating research workflows, similar to BioCoder[3] (which is included by LTS, in an updated C++ version, as one MOSAIC library). Finally, MOSAIC-SR presentations may be based on annotations applied to research/analytic code. For example, in the context of image analysis, the Pandore project (an image-processing application) provides an "Image Processing Objectives" Ontology as well as a suite of image-processing operations that can be called from computer code. Image-analysis pipelines can therefore be explained by annotating the pertinent function-calls (which Pandore calls "operators") with terms from the Pandore controlled vocabulary, providing annotation targets for MOSAIC-SR presentations.

MOSAIC-SR can express both computational workflows that are fully encapsulated by published code and real-world protocols concerning laboratory equipment and physical materials or samples under investigation. In the latter guise, MOSAIC-SR code can employ or instantiate standardized terminologies and data structures for describing experiments — such as MIBBI policies or BioCoder functions. In this case, the role of MOSAIC-SR code is to serve as a serialization/deserialization endpoint for

[2] This toolkit provides a good case-study for research publication because it has an innovative Qt and Common Workflow Language based extension mechanism; cf. https://cbica.github.io/CaPTk/tr_integration.html.

[3] See https://jbioleng.biomedcentral.com/articles/10.1186/1754-1611-4-13.

sharing research metadata. Conversely, when workflows are fully implemented within software developed as part of a body of research, MOSAIC-SR can provide a functional interface allowing this code to be embedded in scientific software. For the latter, MOSAIC-SR provides a framework for modeling how a software component specific to a given research project exposes its functionality to host and/or networked peer applications. There are also instances where both scenarios are relevant — the MOSAIC-SR code would simultaneously document real-world experimental protocols and construct a digital interface as part of a workflow which is part digital ("*in silico*") and part "real-world" ("in the lab").

MOSAIC Annotations

Included in any MOSAIC plugin would be specially-designed PDF viewers for interactively reading authors' papers. In particular, these PDF applications would recognize cross-references linking between publications and their associated supplemental archives. This would allow authors to identify concepts which are discussed and/or represented both in the research paper and in the archive, annotating their papers accordingly. For example, the concept "RNA Extraction" may be discussed in a publication text, and also formally declared as one step in the lab processing as represented via BioCoder, summarized in a BioCoder-generated chart, and included in the supplemental archive (a similar example is used in the documentation for BioCoder). The PDF viewer would then ensure that the phrase "RNA Extraction" in the text is interactively linked to the concordant step in the experimental process, so that readers would then be able to view the BioCoder summary as a context-menu action associated with the phrase where it appears in the PDF file. For a different example, the phrase "Oxygenated Airflow" refers to airflow in assisted-breathing devices, such as ventilators; to ensure that the device is working properly, the equipment must be monitored to check that a steady stream of oxygen reaches the patient. Research into the design and manufacturing of ventilators and similar devices may then include "Oxygenated Airflow" both as a phrase within the article text and as a parameter in data sets evaluating the device's performance. In this situation, the publication-text location of the "Oxygenated Airflow" phrase should once again be annotated with links to the relevant part of the data set (e.g., a table column) where measurements of airflow levels are recorded.

Part II: Concrete Applications

MOSAIC in the context of Image Analysis

This section will focus on one specific application of MOSAIC-SR in the context of image analysis and bioimaging — specifically, what we term D-SPIN ("Data Structure Protocol for Image-Analysis Networking"). D-SPIN adds a narrower focus by extending MOSAIC-SR. Software that uses the D-SPIN protocol is able to provide a description of image processing capabilities which have been utilized and/or are functionally exposed by code and data associated with a research project. This includes "structured reporting" of research objectives as well as a concrete interface for invoking analyses associated with the relevant research (either new algorithms or techniques used to obtain reported findings). D-SPIN, in turn, is based on CaPTk and Pandore (which includes both data models and interactive software) and the Pandore "Image Processing Objectives" Ontology, mentioned above. D-SPIN adopts protocols from CaPTk in order to accept For information about how different objectives are merged into workflows,
— particularly with respect to implementating image-analysis capabilities as extensions to a core application, and with respect to CaPTk's implementation of the Common Workflow Language (CWL). In effect, D-SPIN formalizes the data models and prototypes adopted by Pandore and CaPTk so as to concretize MOSAIC-SR for the specific domain of image processing and Computer Vision. The sections below will therefore outline D-SPIN features in the context of MOSAIC-SR design principles and objectives.

Meta-Procedural Modeling in D-SPIN and MOSAIC-SR

When developed as a systematic outline of computational workflows — not just a digital summary of real-world (e.g. lab) protocols — MOSAIC-SR reports can be formalized using a MOSAIC component which we call "Hypergraph Multi-Application Configuration Language" (HMCL). Most approaches to modeling research workflows involve some concept of "meta-objects",[4] "tools" (in the terminology of CWL), or "transitions" (in the language of Petri Net theory). In HMCL, the equivalent concept is that of *metaprocedures*, which are analogous to ordinary computational procedures but add extra sources of information concerning input and output parameters. In general, rather than simply passing an input value into an executable routine, metaprocedures define steps which can be taken to acquire the proper values when needed. Aside from ordinary runtime values, the most important input sources are methods defined on GUI components; command-line parameters; file contents; and not-yet-evaluated expressions (perhaps encapsulated in scripts or function pointers). A metaprocedure formulation abstracts the acquisition of input arguments (or the consumption of values within "channels" via which a procedure sends and receives data) from the concrete procedure or procedures which are eventually executed. Therefore, an HMCL metaprocedure definition has two separate parts: a preamble where input sources are described; and an executive sequence where concrete procedures are indicated. A *meta-evaluator* then operates in accord with these definitions, concretizing the input values and launching the actual procedure(s). For D-SPIN, metaprocedures can be defined using a framework based on BIOCODER, but adopted to the imaging and Computer Vision context.

Image analysis methods are often described in academic literature in terms of mathematical formulae and/or characterizations of computing environments (such as Graphical Processing Units). It requires additional construction to translate these overviews into actual computer code. Once Computer Vision innovations are in fact concretely implemented, there is then an additional stage of development requisite for users to enact in practice the computations described in the research. Although it is theoretically possible to demonstrate novel methods within fully self-contained autonomous applications, it is more convenient for users if research code is integrated with existing imaging software. Along these lines, the D-SPIN interface can help connect new code to existing applications, allowing users to access the new code's functionality through GUI actions, command-line invocations, or inter-application messaging.

In addition to pragmatically enabling application embedding, D-SPIN models represent research methods and theories, contributing to transparency and reusability according to the MIBBI and FAIR (Findable, Accessible, Interoperable, Reusable) standards.[5] This can be achieved, in part, by implementing data structures conforming to PANDORE Image Processing Objectives. However, D-SPIN embeds this logic in an Object-Oriented context which allows imaging-specific workflow notations to be paired with specifications outside of image-processing in the narrowest sense. This allows D-SPIN to be available for hybrid computational-objectives representations which are only partially covered by the imaging domain — analogous to the MIAPE-GI (Gel Electrophoresis Informatics) component of MIAPE (Minimum Information About a Proteomics Experiment). The following section will discuss several domains where D-SPIN has been explicitly integrated with code libraries codifying MIBBI-style research protocols.

D-SPIN in Contexts Supplemental to Image Processing.

Image Flow Cytometry

One important use-case for biomedical image processing is to analyze cellular microscopy in conjunction with cytometric experiments which indirectly investigate cells and cellular-scale entities (such as proteins). Conceptually, image analysis and flow cytometry (FCM) analysis are mathematically simi-

[4] See the VISSION system: https://pdfs.semanticscholar.org/1ad7/c459dc4f89f87719af1d7a6f30e6f58dff17.pdf.
[5] See https://www.researchgate.net/publication/331775411_FAIRness_in_Biomedical_Data_Discovery.

lar, and some commercial cytometry software has been extended with image-processing capabilities. The overlap between cytometric and image analysis has also inspired attempts to merge cytometry standards, such as MIFlowCyt (the Minimum Information about a Flow Cytometry Experiment policy within MIBBI), with bioimaging standards such as DICOM (Digital Imaging and Communications in Medicine). One such proposal is due to Robert Leif, who argues that "The large overlap between imaging and flow cytometry provides strong evidence that both modalities should be covered by the same standard" and has formalized an XML language (CytometryML) to serve as that overarching bridge.[6] The D-SPIN project builds off this work by introducing its own FCM/DICOM hybrid, although in an object-oriented rather than XML-based context, although it also incoporates an expanded XML-oriented schema (discussed in the next part of this paper). As a reference implementation for this D-SPIN extension, the project also provides a pure-C++ cytometry library based on the openCyto and FACSanadu libraries, while eliminating external dependencies such as R and Java.[7] The FCM/DICOM bridge is implemented in this context via a D-SPIN supplement to DICOM based on "Semantic DICOM," which is an effort to standardize query processing within PACS (Picture Archiving and Communications Service) workstations and to more effectively integrate DICOM with clinical data.

A Semantic DICOM Object Model

As a formal representation of imaging workflows, D-SPIN would reasonably be paired in many contexts with DICOM, insofar as DICOM represents the canonical standard for exchanging medical image data. For its applications within the medical-imaging context, D-SPIN, therefore, provides object-oriented accessors to DICOM data such that image-objectives and DICOM object models can interoperate. This object-oriented foundation also provides a basis on which to further integrate clinical data in the form of "Semantic" PACS models.[8] The original Semantic PACS implementation draws clinical data from DICOM headers, and represents this extracted information via RDF. In keeping with MOSAIC's more object-oriented focus, the MOSAIC-Semantic DICOM (MOSAIC-SD) integration is engineered instead as an extension to the DICOM Toolkit (DCMTK) library, though it imports many constructs within Semantic DICOM in the guise of a "Hypergraph Ontology" — one example of a system MOSAIC uses to merge Semantic Web schema with Object-Oriented code. In short, MOSAIC-SD extends D-SPIN by defining a central DICOM object model affixed to both clinical and image-processing object models.

The MOSAIC-SD object system applies not only to DICOM integrated with statements of image-processing objectives, but also to other biomedical contexts where image analysis should be integrated with other analytic modalities as well as with clinical or epidemiological information. For example, Flow Cytometry overlaps with clinical data tracking because one of FCM's essential investigative roles is to examine patients' immunological response to diseases and/or interventions. In the context of Covid-19, say — with respect to achieving a deeper understanding of how and why SARS-CoV-2 symptoms present differently in different patients — "The starting point will likely be a deep characterization of the immune system in patients with different stages of the disease."[9] That is, FCM observations need to be matched with clinical data in order to classify (and consider statistical correlations between) immunological findings and clinical facts: risk factors, sociodemographics, disease progression, and so forth. This analysis intrinsically assumes that FCM data can be transparently linked with all relevant clinical data, but such integration is difficult even in DICOM, where DICOM headers are specifically designed for preserving patient data across picture-sharing networks (there is no analogous "header" component in the Flow Cytometry Standard, FCS). In brief, D-SPIN

[6] See https://spie.org/Publications/Proceedings/Paper/10.1117/12.2295220?SSO=1.

[7] Upon request, LTS can provide a detailed summary of this project reconciling openCyto and FACSanadu. Specifically, although openCyto as a whole uses R for its visual layer, openCyto contains cytoLib, a C++ library for cytometric analysis, which can be used as the basis for a standalone pure-C++ flow cytometry application. Meanwhile, FACSanadu is a Java application which uses a Qt front-end that we are migrating to C++. The supplemental information about the FACSanadu/cytoLib integration identifies the specific data types where cytoLib code (different kinds of gates, for instances) can be incorporated as alternatives to the FACSanadu Java equivalents, while still using FACSanadu GUI classes.

[8] See https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5119276/.

[9] See https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.24002.

extensions to non-imaging domains can promote data integration insofar as D-SPIN Clinical Object Models, based on Semantic DICOM, provide an affixation point for clinical data in analytic contexts which are operationally related to image analysis, and not just to image analysis by itself.

Semantic DICOM via DCMTK Extensions

As a concrete example of how a Semantic DICOM Object Model could be implemented, consider the process of developing DICOM extensions by modifying existing DICOM client libraries, such as DCMTK. As a file *standard*, DICOM itself is an abstraction; the DICOM protocol only becomes a concrete phenomenon insofar as PACS workstations, radiology software, and other applications employ DICOM client libraries. Therefore, any DICOM *extension* is similarly concretized only by an application which links against modified DICOM libraries that recognize the extended syntax and/or semantics. When implementing DICOM, accordingly, we can assume that the modified client libraries exist in an ambient context which provides capabilities that can be leveraged by the extension as it becomes formalized. For example, if a DICOM extension provides a unified FCM/DICOM format, we can assume that the workstation linking against modified DCMTK libraries supports FCM features, such as the ability to construct geometric gating models (that is, rectangular or ellipsoid segments on FCM images) and to consume information about the cytometric equipment used to derive the current FCM data.

In the case of Semantic DICOM, we can similarly assume that the host application which loads an enhanced "Semantic" version of a DICOM client library has the ability to consume more complex clinical data models than are incorporated into conventional DICOM. In the case of DCMTK, patient information is accessed through a **DcmDataset** object, which in turn contains multiple **DcmObject** values spanning several more specific types. A DICOM extension can, accordingly, be concretized by expanding the range of object types subsumed under **DcmObject**, as well as modifying the **DcmDataset** code so that instances of these extended types could be identified and passed off to the appropriate handlers. Developers can then bundle files serializing these extended types within the overall collection of files zipped into a single DICOM resource.

The steps outlined above allow an extension object model to be embedded in DICOM by hooking extra processing code into the procedures whereby conventional **DcmDataset**s are extracted from DICOM files. It is then necessary to ensure that the DICOM client application can properly manipulate the extension object data. This could be handled simply by linking all necessary libraries into the DICOM client itself, but a more flexible solution is to employ some form of multi-application networking. In the case of non-standard clinical data, a flexible approach would be to pair client libraries for serializations of such data with a standalone application that can parse these serializations and respond to requests about the encoded information. With such an application in place, the DCMTK host application would not need to implement all logic related to the extension object model; it would simply need to be able to launch and/or communicate with a secondary application tailored to each individual extension in particular.

As a concrete example, consider a specialized data model and self-contained application specifically devoted to tracking immunological responses and post-recovery symptomology for Covid-19. The goal would be to develop more sophisticated technology for modeling the progression of active Covid-19 infections, as well as the lingering effects of the disease for patients who experience adverse reactions (such as lasting neurological damage or lung impairment) even after their ostensive recovery. Detailed models of Covid-19 affliction could then be matched against epidemiological, treatment, and sociodemographic data to determine whether there are predictors that determine which patients may experience more severe or long-lasting symptoms, and whether these symptoms can be mitigated by different treatment options. The data consumed by such a Covid-specific application might be bundled into DICOM files even if DICOM clients cannot directly process this data. In such a case, they could, instead, be programmed to launch the Covid-specific application and query that application for select pieces of information that *are* relevant for a DICOM workstation (e.g., an immunological profile of the patient adding layers of detail to cytological imaging). In short, a Covid-specific application would potentially be accessed by several different peer applications ad-

dressing several different biomedical domains (bioimaging, cytometry, epidemiology, genomics), and could provide different sorts of information from a Covid-specific data model for different contexts: given an overall information package about a patient's SARS-CoV-2 immunology and symptomology, some categories of data will be relevant for bioimaging, while others will be relevant for genomics and so forth. The file formats employed by these various peer applications (such as **DICOM**, in the bioimaging context) may then be extended only as much as is needed to seed shared data packages with enough information to launch the Covid-specific application and request information specific to the peer application's specific biomedical domain.

Geo-Imaging and Geographic Information Systems

Another area where **D-SPIN** provides structured object models is that of Geographic Information Systems (**GIS**) annotations. There is a direct link between image processing and **GIS** insofar as identifying geotaggable features is one dimension or application of Computer Vision. Effectively manipulating geoimaging data requires mathematical translations between several different coordinate systems, in both two and three dimensions. These coordinate transforms — as well as semantic interpretations of geoimage segments (buildings, land features, roads, etc.) — can serve as the basis for an object model attaching image-processing objectives to **GIS** workflows.

In conventional **GIS** annotation, data structures are linked to both geospatial coordinates and to visual cues or icons allowing locations of interest to be indicated on maps. The actual geotagged data structures could be derived from any domain; as such any object model may be integrated with **GIS** annotations so long as one can assign spatial interpretations to the phenomena computationally encapsulated by the domain in question. In a medical context, for instance, geotagged data might represent the scope of a vaccination campaign, or the extent of an epidemic, along with relevant geographic or civic features (villages, medical clinics, national borders, and so on). The actual map as a visible digital artifact therefore serves as a virtual glue where clinical, geographical, governmental, and **GUI** data are all sutured together. Insofar as geoimaging involves analysis of photographed land features and/or urban environments, image processing information represents a further object model that can be added to the mix. Even when dealing solely with virtual maps, however — rather than with satellite images or other geospatial photography — the analysis and application-level rendering of map features is sufficiently similar to image processing that a rigorous model of **GIS** integration belongs properly within **D-SPIN**, specifically within a **MOSAIC**-GIS extension.

Part III: MOSAIC Data Models

Hypergraph Database Models for Publication Data

At the core of any **MOSAIC** portal is a collection of distinct files, representing individual publications and supplemental archives. However, in a typical case it will be necessary to ground the system of files in a database hosting publication information, so that readers could search the portal for specific authors, keywords/phrases, subject areas, and so forth. Searches within publication metadata are the simplest example of this functionality. Searches within publication texts themselves are more complex, because what seems like a straightforward keyword search from a user's point of view may be complicated at the processing level by documents' markup structure — technical terms or acronyms which readers see as single words/phrases could actually be defined within the document by internally structured text segments, rather than raw character streams. Keyword searches might also be tripped up by ligatures, footnote interruptions, and other visual features wherein the document in human-readable formats like **PDF** diverges from machine-readable raw text.

To address these potential search complications, **MOSAIC** provides a "Hypergraph Text Encoding Protocol" (**HTXN**) which separates document text into different layers, dividing presentation-related content from textual content and making search capabilities more reliable. Each publication can have an **HTXN** representation, essentially a machine-readable structure encapsulating documtn text,

as part of its supplemental archive. These HTXN resources can then be used to support robust searching among publication sets.

Apart from metadata and keyword searches, an additional layer of search functionality within MOSAIC portals is the option to search across supplemental archive contents such as data sets and research protocol descriptions. MOSAIC provides a hypergraph-based database engine that can be used, at a minimum, for publication info; developers have the option of including supplemental content as well within this central database. Each supplemental archive encompasses raw data and/or methodological descriptions via files internal to the archive; these assets are not necessarily shared with the host MOSAIC repository except by directly examining archive files. However, a repository could choose to model some or all archive content within its publication database, either directly storing archives' data sets or importing certain select information (such as object/type schema or representative data samples). In this case, some data in the supplemental archives will be mirrored in the central database, so that it is accessible for user-facing searches. MOSAIC employs a flexible Hypergraph Database engine which can accommodate many different kinds of data models, from "SQL-like" tables to graphs and hierarchical collections types. Therefore, the publication database within a MOSAIC portal would have the flexibility to import most data which is present in supplemental content associated with given publications.

In some contexts, authors may choose specifically to structure their shared research data in a format that can be exported to a MOSAIC publication database. This is because MOSAIC's hypergraph data model supplies a rigorous framework for documenting how a data set is organized, which in turn can provide a formal overview of the theoretical or methodological commitments informing the research. The MOSAIC database engine — called ConceptsDB — represents an expanded hypergraph metamodel built around different paradigms for expressing scientific knowledge, such as Conceptual Space Theory and Conceptual Role Semantics.[10]

Markup Serialization and "Grounding"

The data-integration mechanisms discussed in this paper thus far have focused on object models and object-oriented programming techniques. While composing special-purpose object-based libraries specifically tailored to individual data-integration problems is a powerful tool for solving such problems, data integration initiatives in practice are often organized around standards for sharing or serializing conformant data structures. As a result, an important aspect of data integration is implementing proper data-serialization technology which is sufficiently rigorous to serve as a proxy for formal interface definitions. XML formats are a good case-study for such formalizations because most serialization in the biomedical and bioimaging context operates through XML languages.

Standardizing a data format by stipulating how XML files may encode the data is simpler than defining an analogous specification in terms of executable computer code: one way to document the shape of any relevant data is just to explain how an XML document will be structured insofar as it encodes data accordingly. Potentially, such specification can be a single XML DTD file, or an XML sample, providing a convenient reference point for developers to grasp the underlying data model. However, the structure of an XML document does not, in itself, present a clear picture of how the information which the document represents is semantically organized. Even though XML is processed by computer programs, it is not even evident from an XML document or schema which XML elements (if any) correspond to data types recognized by applications which read and/or write the corresponding XML code. For example, the XML portion of OME-TIFF (the principal Open Microscopy Environment imaging format) includes an explicit **Image** element (which gathers up all significant image metadata); an application reading OME-TIFF files might therefore introduce a single datatype — analogous to (and maybe wrapping an) **itk::Image** from the Insight Toolkit imaging libraries — bundling the data in that part of the OME XML. In this case there will be a one-to-one correspondence

[10] Or at least a variation of Conceptual Roles consistent with AI tools such as GRAKEN.AI, and formalizations of Conceptual Spaces such as those described in https://arxiv.org/abs/1706.06366.

between XML structure and application-level data types, at least for that one **Image** node. On the other hand, software reading OME-TIFF information might not manipulate images directly, but rather pull out other kinds of metadata, such as an experimenter's name or description of the microscopic setup. In this case, the application might not have an explicit "object" representing the image itself, but it may still read information about the image from child nodes of the XML **Image** element.

In short, applications can read or use data from an XML document in different ways, so the document's structure does not itself provide a clear picture of how code which reads the XML is organized. This uncertainty is significant insofar as one wishes to use XML specifications as an indirect strategy for documenting parameters and features of the data structures which are serialized via the relevant XML language. In effect, XML serialization operates on two levels: on the one hand, the specific XML document provides an encoding of data conforming to a given structure; but, at a more abstract level, one can model the relationship between the surface-level XML node structure and the application-level data structures thereby serialized. This second level of detail is usually implicit and unstated, but in MOSAIC, such "meta-serialization" — a term used to suggest the idea of providing meta-data *about* a serialization — is directly formulated through a notion of "grounding", which involves adding supplemental markup clarifying how documents instantiating a serialization format (such as XML) relate to the coding protocols and data types of software which reads or creates these documents.

For maximum expressivity, MOSAIC introduces a meta-serialization system that can be applied to languages more flexible than XML — notably TAGML — as well as to XML proper.[11] In effect, MOSAIC provides parsers for an extended version of TAGML which includes an additional "grounding" layer. Grounding, in this context, means describing how elements in the markup — nodes, attributes, and character sequences — are "grounded" in application-level types, data fields, and other programming constructs. MOSAIC provides parsers for this "Grounded TAGML" (GTAGML) language as well as converters to output serialized data as conventional XML, so that GTAGML specifications can be used in contexts (MIFLOWCYT, for instance) where ordinary XML is expected and serves as a basis of standardization. MOSAIC also provides code libraries for extracting data from GTAGML documents.

GTAGML documents, with certain restrictions enforced, are structurally isomorphic to XML and can be rendered as pure XML; as such, GTAGML schemata can be used to define norms for XML languages, although the logical role and operations of such requirements is not identical to XML schema definitions. By design, GTAGML schemata operate on two levels: on the one hand, they constrain the organization of GTAGML files themselves; but they also stipulate how GTAGML document structure relates to the type systems of code libraries serializing and deserializing GTAGML files. To model this second, type-oriented metadata, GTAGML introduces a hypergraph-based type model organized around "infosets," which are structured overviews of application-level data types. To fully utilize GTAGML features, programmers may then compose "infoset classes" as wrappers around ordinary data types (e.g., C++ classes) whose instances are serialized via nodes or character strings within GTAGML documents. Infoset classes provide hypergraph "views" onto type-instances, and act as a bridge between data types and their associated GTAGML schema. In particular, infoset classes (rather than the types they encapsulate) are the basis for schematizing the relation between GTAGML document elements and type instances (and the data they contain).

The hypergraph-based modeling incorporated into GTAGML reuses much of the code associated with ConceptsDB, which is the new hypergraph database being developed alongside MOSAIC. More information about ConceptsDB — and on the HTXN text encoding protocol for MOSAIC portal manuscripts, mentioned earlier — can be found on the guthub project page for Linguistic Technology Systems (https://github.com/Mosaic-DigammaDB/LingTechSys).

For more information please contact:
Amy Neustein, Ph.D., Founder and CEO
Linguistic Technology Systems
amy.neustein@verizon.net • (917) 817-2184

[11] See http://www.balisage.net/Proceedings/vol21/print/HaentjensDekker01/BalisageVol21-HaentjensDekker01.html.

Appendix: Integration between Different Flow Cytometry Libraries

As discussed above, one use-case for data integration via D-SPIN is to merge the features of multiple FCM libraries, such as CYTOLIB and FACSANADU (although the techniques discussed here could be applied to other libraries as well). In the CYTOLIB/FACSANADU reconciliation, FACSANADU code (translated from JAVA to C++) provides GUI logic whereas CYTOLIB provides most of the analytics and FCS processing. These two code bases come into contact insofar as CYTOLIB data needs to be visualized within FACSANADU GUI components. Constructing a bridge between the two systems therefore involves translating CYTOLIB data structures to the data types expected by FACSANADU when it initializes visual renderings of FCS data, such as event plots and gate boundaries.

To facilitate this translation, the D-SPIN-based strategy involves isolating certain geometric and data-modeling primitives and implementing them in a separate code base distinct from either of the FCM components. These modeling primitives then as an intermediary between CYTOLIB and FACSANADU, but they are also designed with an eye toward being applicable in other image-processing contexts. Important data structures involved in this bridge code include the following:

Vertex Sets The important feature of geometric vertices in the FCM context is that vertices need to be expressed within and mapped between several different coordinate systems. For example, CYTOLIB uses a different "coordinate" class when reading data via Protocol Buffers than for internally defining gates. At the GUI level, at least three different notions of two-dimensional locations are important (and require transforms such as **mapFcsToScreen** in FACSANADU) depending on whether one is tracking visual objects — or events, such as mouse clicks — relative to the entire computer screen, or one GUI window, or the top-left corner of the object whose method handles a given user-interaction event. A further complicating factor is that some coordinate systems use integer values, while others using floating-point numbers. In other contexts related to FCM, such as PDF rendering and image analysis, there are still further coordinate systems; so it is not uncommon for a code base (even just in the 2D context) to manage spatial positions recorded in at least seven different formats. It is therefore a resonable practice to factor out different coordinate systems into a common 2D position/vertex model from which context-specific locations can be derived.

Ellipses Different applications represent ellipses in different ways, including via antipodal points, axis radii, or covariance matrices. The matrix form (which entails four values, or 2×2, for a two-dimensional shape) is preferred for FCM (as standardized in GATING-ML, say), but other representations are more common in other contexts, such as image analysis (, for instance, distinguishes "object space" from "world-space", which has the effect of separating a two-valued description of the elliptical shape as an intrinsic morpholigical quality, whereas the ellipse's axis angles are a feature of how its intrinsic space relates to a parent space). In QT, for its part, ellipses (a kind of **QGraphicsItem**) are defined in terms of a bounding rectangle together with a rotational transform. Given this diversity of representations, D-SPIN uses a separate ellipse class (based on eccentricity and major-axis angles) which can be mathematically transformed into these different formats.

Gate Transformations Only linear and logarithmic transformations are defined in FACSANADU, although it is straightforward to further the subclass the **Transformation** base type. To be consistent with the GUI, modified transformation algorithms need to be registered with a context menu in "**ViewWidget**" (a menu activated by clicking in a boundary area, outside any rendered data). This is an example of where data modeling/analytic features and GUI setup needs to be correlated, which D-SPIN approaches via structured GUI modeling and annotations (see the MOSAIC Annotations section above). A wider inventory of transformation strategies are implemented in CYTOLIB, such as biexponentials and "logicles," but there are other transformation methods proposed in academic literature which are not implemented for CYTOLIB. The D-SPIN approach is to prioritize extensibility, with a transformation class structured to facilitate both subclassing *and* GUI integration. This is part of a more general design philosophy for image-processing algorithms where code extensions can be deployed and annotated to demonstrate novel analytic techniques described in scientific literature, with microcitation-based cross-references between code, publications, and GUIs.