



The **MOSAIC** Structured Reporting Framework and Image Processing Objectives Description

The “**MOSAIC** Data-Set Explorer” (**MdsX**) and “**MOSAIC** Structured Reporting” (**MOSAIC-SR**) are tools to help authors develop interactive presentations supplementing academic documents (**MOSAIC** is an acronym for “Multi-Paradigm Ontologies for Scientific and Technical Publications”). With **MdsX**, interactive presentations take the form of software applications that provide access to data sets, analytic techniques, or other digitally representable artifacts to document or encapsulate research work. With **MOSAIC-SR**, authors can implement or reuse code libraries that report on research/experiment methods, workflows, and protocols. Conceptually, **MOSAIC-SR** is functionally similar to the various domain-specific recommendations collectively gathered into the “Minimum Information for Biological and Biomedical Investigations” (**MIBBI**) specifications, and indeed one use-case for **MOSAIC-SR** is that of implementing object models instantiating **MIBBI** policies. In some contexts, **MOSAIC-SR** and **MIBBI** overlap, because elements of scientific workflows are sometimes algorithms implemented within a code package concretizing authors’ research.

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 **BIOCORDER** functions.¹ In this case, the role of **MOSAIC-SR** code is to serve as a serialization/deserialization endpoint for 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 these latter cases, **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 cases 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”).

This paper will focus on one specific application of **MOSAIC-SR** in the context of image analysis and bioimaging — specifically, a “Data Structure Protocol for Image-Analysis Networking” (**D-SPIN**), which both extends and adds a narrower focus to **MOSAIC-SR**. Software using the **D-SPIN** protocol provides 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 research (either new algorithms or techniques used to obtain reported findings). **D-SPIN**, in turn, is based on **CAPTK** (the Cancer Imaging and Phenomics Toolkit) and **PANDORE** (an image-processing environment which includes both data models and interactive software). The **PANDORE** project encompasses an ontology of “Image Processing Objectives” that provides a structural basis for **D-SPIN**. For information about how different objectives are merged into workflows, **D-SPIN** adopts protocols from **CAPTK**, particularly with respect to implementing 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 following sections will therefore outline **D-SPIN** features in the context of **MOSAIC-SR** design principles and objectives.

¹See <https://jbioleng.biomedcentral.com/articles/10.1186/1754-1611-4-13>.

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

Most approaches to modeling research workflows involve some concept of “meta-objects”,² “tools” (in the terminology of **CWL**), or “transitions” (in the language of Petri Net theory). In **MOSAIC-SR**, the analogous concept is that of *meta-procedures*, 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 meta-procedure 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, a **MOSAIC** meta-procedure 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**, meta-procedures can be defined using a framework based on **BIOCORDER**, 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 actually enact 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 practically 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.³ 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 **MAPE-GI** (Gel Electrophoresis Informatics) component of **MAPE** (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 investigate cells and cellular-scale entities (such as proteins) indirectly. Conceptually, image analysis and flow cytometry (**FCM**) analysis are mathematically similar, and some commercial cytometry software has been extended with image-processing capabilities. The overlap between cytometric and image analysis has also inspired

²See the **VISSION** system: <https://pdfs.semanticscholar.org/1ad7/c459dc4f89f87719af1d7a6f30e6f58dff17.pdf>.

³See https://www.researchgate.net/publication/331775411_FAIRness_in_Biomedical_Data_Discovery.



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.⁴ 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 (along with an expanded **XML**-oriented schema discussed in the next section). 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, but eliminating external dependencies such as **R** and **JAVA**. 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, therefore, **D-SPIN** 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.⁵ 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 and also 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 stage of the disease".⁶ 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** 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, not just to image analysis per se.

⁴See <https://spie.org/Publications/Proceedings/Paper/10.1117/12.2295220?SSO=1>.

⁵See <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5119276/>.

⁶See <https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.24002>.



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 can be derived from any domain; as such any object model can 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 may 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.

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 between **XML** structure and application-level data types, at least for that one **Image** node. On the other hand, software reading **OME-TIFF** information



may 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 may 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.⁷ 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 a new hypergraph database being developed alongside **MOSAIC**. More information about **ConceptsDB** can be found on the github 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

⁷See <http://www.balisage.net/Proceedings/vol21/print/HaentjensDekker01/BalisageVol21-HaentjensDekker01.html>.

