WHITE PAPER

> ## The MOSAIC Publication Repository and Structured Research Reporting Framework

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 author's name, keywords, or subject matter, and a document viewer where readers can view each publication's abstract (and sometimes its full text) and other publication details (such as co-authors, dates, citations, 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.
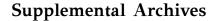
The "MOSAIC" system is a suite of code libraries allowing institutions to host publication repositories which address the limitations of existing scientific document portals (MOSAIC is an acronym for "Multi-Paradigm Ontologies for Scientific and Technical Publications"). With MOSAIC, each publication can be linked to a "Supplemental Archive" which contains information about the author's research methods, data sets, and focal topics. Using MOSAIC, developers can implement a hosting platform for this supplemental material, in addition to publications themselves, where the platform provide 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 be general-purpose portals hosting academic or technical papers covering a broad range of topics; or narrower collections of articles focused on specific subject areas or institutional archives. Organizations can use use MOSAIC internally as the basis of a document management system (**DMS**). Conversely, MOSAIC could be used to provide open public access to complete publications, including human-readable and machine-readable full text and all supplemental content, with no restrictions[1] (any project which *does* restrict access to some part of any document requires a commercial MOSAIC license). MOSAIC can also 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 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 as scientists use to conduct research. MOSAIC also introduces a "structured reporting system" (MOSAIC-SR) for describing research/experimental/lab methods and protocols. Supplemental Archives, plugins, and MOSAIC-SR are outlined in the following sections.

---

[1] Excepting sensitive/private information which might be provided via a dedicated component within the portal with specific features for authors, editors, and administrators

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. Structured files containing raw data discussed in the publication, along with interactive software allowing scientists to access and reuse the data;

3. 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;

4. Representations of analytic methods and algorithms used to derive the research findings, provided directly via computer code or indirectly via formal descriptions of computational workflows;

5. Self-contained computer software demonstrating code which the author developed and/or used for analyzing research data;

6. 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 raw data. In the former case, a typical archive will include a 1data-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 needing to be serialized for a given publication. The resulting data-set application can then be packaged as a plugin to existing scientific software, depending on the relevant scientific discipline to which the research belongs.

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, 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 actually 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 setup, as a result, the software ecosystem wherein readers examine published research is

fundamentally separate from the software platforms where research is actually conducted. This is one example of how existing portals are limited in their ability to share research in rigorously reusable and replicable ways, and how MOSAIC offers a potential improvement.

For authors, embedding presentations of their research within existing scientific software has the added benefit of making their work more practically and immediately useful for the scientific community. Such presentations establish the computational framework for practically 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. This kind of functional assessment and adoption of novel contributions is harder to carry out if a research work is only described indirectly within article text, as opposed to being concretely developed for execution 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, say, 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, so the scientific software is a better forum for showcasing novel research.

MOSAIC Structured Reporting

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 thier 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[2] (which is included, 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 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.

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 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

---

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

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").

MOSAIC Annotations

Included in any MOSAIC plugin would be specially-designed PDF viewers for interactively reading student's papers. In particular, these PDF applications would recognize cross-references linking between student publications and their associated Supplemental Archives. This would allow readers to identify concepts which are discussed and/or represented both in the research paper and in the archive, annotating their papers accordingly. For example (a similar example is used in the documentation for BioCoder), 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, and summarized in a BioCoder-generated chart (and included in the Supplemental Archive). 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 could 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 ensure 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 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, 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 "Image Processing Objectives" Ontology, mentioned above, 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 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 following sections 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

Most approaches to modeling research workflows involve some concept of "meta-objects",[3] "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 imput 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 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 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.[4] 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 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

---

[3]See the VISSION system: https://pdfs.semanticscholar.org/1ad7/c459dc4f89f87719af1d7a6f30e6f58dff17.pdf.
[4]See 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.[5] 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.[6] 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".[7] 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.

---

[5]See https://spie.org/Publications/Proceedings/Paper/10.1117/12.2295220?SSO=1.
[6]See https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5119276/.
[7]See https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.24002.

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 a 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 which loads an enhanced, "Semantic" version of a DICOM client library has the ability to consume more complex clinical data 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 then 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 can 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 just outlined 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 in to 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 the extensions 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 lingering effects of the disease for patients who experience adverse reactions (such as lasting neurological damage) 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 for which patients may experience more severe or long-lasting symptoms, and whether these can be mitigated by different treatment options. The data consumed by such a Covid-specific application could be bundled into DICOM files even if DICOM clients cannot directly process this data; instead, they could 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 could be accessed by several different peer applications addressing 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, others for genomics, and so forth. The file formats employed by these various peer applications (such as **DICOM**, in the bioimaging context) could 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 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.

Part III: 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.[8] 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 markip — 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).

---

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

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** 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