Proposal for Radiological Software Extensions to Accelerate Patient-Centered Research for Covid-19

Introduction

Patient-centered research in the radiological context focuses on improving the precision of diagnostic-imaging techniques and corresponding clinical interventions. Indeed, the goal of contemporary radiology is not only to confirm a diagnosis, but also to extract cues from medical images that suggest which course of treatment has the highest probability of favorable treatment outcomes. A related goal is curating collections of diagnostic images so as to improve our ability to identify such diagnostic cues, potentially using Machine Learning and/or Artificial Intelligence applied to large-scale image repositories.

The goal of building "searchable" image repositories has inspired projects such as the Semantic Dicom Ontology (SEDI)[1] and the VISION "structured reporting" system.[2] As explained in the context of SEDI: "if a user has a CT scan, and wants to retrieve the [corresponding] radiation treatment plan ... he has to search for the RTSTRUCT object based on the specific CT scan, and from there search for the RTPLAN object based on the RTSTRUCT object. This is an inefficient operation because all RTSTRUCT [and] RTPLAN files for the patient need to be processed to find the correct treatment plan."[3] Even relatively simple queries such as "display all patients with a bronchial carcinoma bigger than 50 cm^3" cannot be processed by PACS systems: "although there are various powerful clinical applications to process image data and image data series to create significant clinical analyses, none of these analytic results can be merged with the clinical data of a single patient."[4] These limitations partly reflect the logistics of how information is transferred between clinical institutions and radiology labs. In response, and in an effort to advance the science of diagnostic image-analysis, organizations such as the Radiological Society of North America (RSNA) have curated open-access data sets encompassing medical images as well as image-annotations (encoding feature vectors) that can serve as reference sets and test corpora for investigating analytic methods. Such repositories are designed to integrate data from multiple hospitals and multiple laboratories — bypassing the conventional data flows wherein radiological information is shared between clinicians and radiologists, but is not also merged into broad-spectrum corpora.

---

[1] See https://bioportal.bioontology.org/ontologies/SEDI.
[2] See https://epos.myesr.org/esr/viewing/index.php?module=viewing_poster&task=&pi=155548.
[3] See https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5119276/
[4] See https://semantic-dicom.com/starting-page/.

This renewed focus on patient outcomes has important consequences for the scope and requirements of diagnostic-imaging software. In particular, the domain of radiological applications is no longer limited to PACS workstations where pathologists perform their diagnostic analysis, with the results transferred back to the referring institution (and subsequently available only through that institution's medical records, if at all). In the older, conventional workflow, radiographic images are requested by some medical institution for diagnostic purposes. Relevant information is therefore shared between two end-points: the institution which prescribes a diagnostic evaluation and the radiologist or laboratory which analyzes the resulting images. Building radiographic data repositories complicates this workflow because a third entity becomes involved — the organization responsible for aggregating images and analyses is generally distinct from both the prescribing institution and the radiologists themselves. As a result, both radiologists and prescribing institutions, upon participation in the formation of the target repository, must identify which image series and which patient data are proper candidates for the relevant repository.

For a concrete example, RSNA has announced the forthcoming publication of an open-access image repository devoted to Covid-19.[5] This repository is being curated in collaboration with multiple European, Asian, and South American organizations so as to collect data from hospitals treating Covid-19 patients. Such a collaboration requires protocols both for data submission and for patient privacy and security. As this example demonstrates, these kinds of data-sharing initiatives present new requirements for radiological software, which must not only allow for the presentation, annotation, and analysis of medical images, but also for participation in data-sharing initiatives adhering to rigorous modeling and operational protocols.

Simultaneously, the science of diagnostic imaging is also expanding as new image-analytic techniques prove to be effective at detecting signals within image data, often complementing the work of human radiologists. The proliferation of image-analysis methodologies places a new emphasis on *extensibility*, where radiological software becomes more powerful and flexible because new analytic modules may be plugged in to a central PACS system. A good example of this new paradigm is the Cancer Imaging Phenomics Toolkit (CaPTk), developed by the Center for Biomedical Image Computing and Analytics (CBICA) at the University of Pennsylvania's Perelman School of Medicine. The CaPTk project provides a central application which supplies a centralized User Interface and takes responsibility for acquiring and loading radiographic images. The CaPTk core application is then paired with multiple "peer" applications which can be launched from CaPTk's main window, each peer focused on implementing specific algorithms so as to transform and/or to extract feature vectors from images sent between CaPTk and its plugins.

Both the patient-outcomes focus in building image repositories and the integration of novel Computer Vision algorithms depend, at their core, on rigorous data sharing. Taking the RSNA Covid-19 repository as a case study for promoting research into post-diagnostic outcomes, this repository is

[5] See https://www.rsna.org/covid-19.

possible because an international team of hospitals and institutions have agreed to pool radiological data relevant to SARS-CoV-2 infection according to a common protocol. Taking CaPTk as a case-study in multi-modal image analysis, this system is likewise possible because analytic modules can be wrapped into a plugin mechanism which allows many different algorithms to be bundled into a common software platform. Of course, these two areas of data-sharing overlap: one mission of repositories such as the RSNA's is to permit many different analyses to be performed on the common image assets. The results of these analyses then become additional information which enlarges the repository proportionately. If CaPTk modules are used to analyze the RSNA Covid-19 images, for example, there needs to be a mechanism for exporting the resulting data outside the CaPTk system, so that the analyses may be integrated into the repository either directly or as a supplemental resource.

This example demonstrates how software such as CaPTk may be extended to support the curation of image repositories dedicated to Patient Outcomes and Comparative Effectiveness Research (CER), insofar as analytic data generated by CaPTk components can acquire the capability to share data according to repository protocols. A further level of integration between CaPTk and CER initiatives can be achieved if one observes that clinical outcomes may be part of the analytic parameters used by CaPTk modules. As presently constituted, CaPTk analytic tools are focused on extracting quantitative (or quantifiable) features from image themselves, without considering additional patient-centered context. There is no technical limitation, however, which would prevent the CaPTk system from sharing more detailed clinical information with its modules, allowing these analytic components to cross-reference image features with clinical or patient information. Insofar as the image analysis is often retroactive — not entertained in the course of a present diagnosis but examining images from which a diagnosis has already been rendered — information about treatment protocols and outcomes can also be shared between the CaPTk components, assuming this information is provided along with images themselves in the context of an image repository and/or a "semantic" DICOM system.

We propose, therefore, to implement enhancements to CaPTk allowing for clinical and outcomes data to be shared between CaPTk components, and allowing for CaPTk modules to participate in data-sharing initiatives devoted to integrating image analysis with outcomes research. Moreover, we believe that the data-sharing protocol used internally by CaPTk can be formalized and generalized to serve as a prototype for integrating diagnostic imaging with clinical outcomes in broader contexts. The protocols adopted by CaPTk reflect how the CaPTk system is designed: each CaPTk component is a semi-autonomous software application providing specific analytic capabilities. By default, each CaPTk component is a *stand-alone* and *native* application which is operationally independent of CaPTk, apart from receiving image data from the main CaPTk application. Integration between CaPTk and its peer applications therefore operates on several different levels, including that of building and compiling the applications alongside CaPTk and registering the modules with the central system, so that users can launch the peer application while running the main CaPTk program. A variation of these protocols applies to components deployed by some means other than source-code level

inclusion in a CaPTk system; e.g. as pre-built binaries or as Docker images.[6] In each of these cases, a rigorous formalization of the CaPTk protocol has to consider compiler, executable, and workflow-related integration between the components, not only shared data formats or Common Vocabularies.

The following sections will describe our proposed contributions to patient-centered research, focusing on the diagnostic-imaging context, via (1) formalizing and enhancing the CaPTk protocol; (2) implementing a general-purpose research platform oriented toward integrating radiological and clinical-outcomes data; and (3) specifying and providing implementations for novel data and workflow languages which we claim are more rigorous, more expressive, and more conducive to application integration than (for example) RDF ontologies.

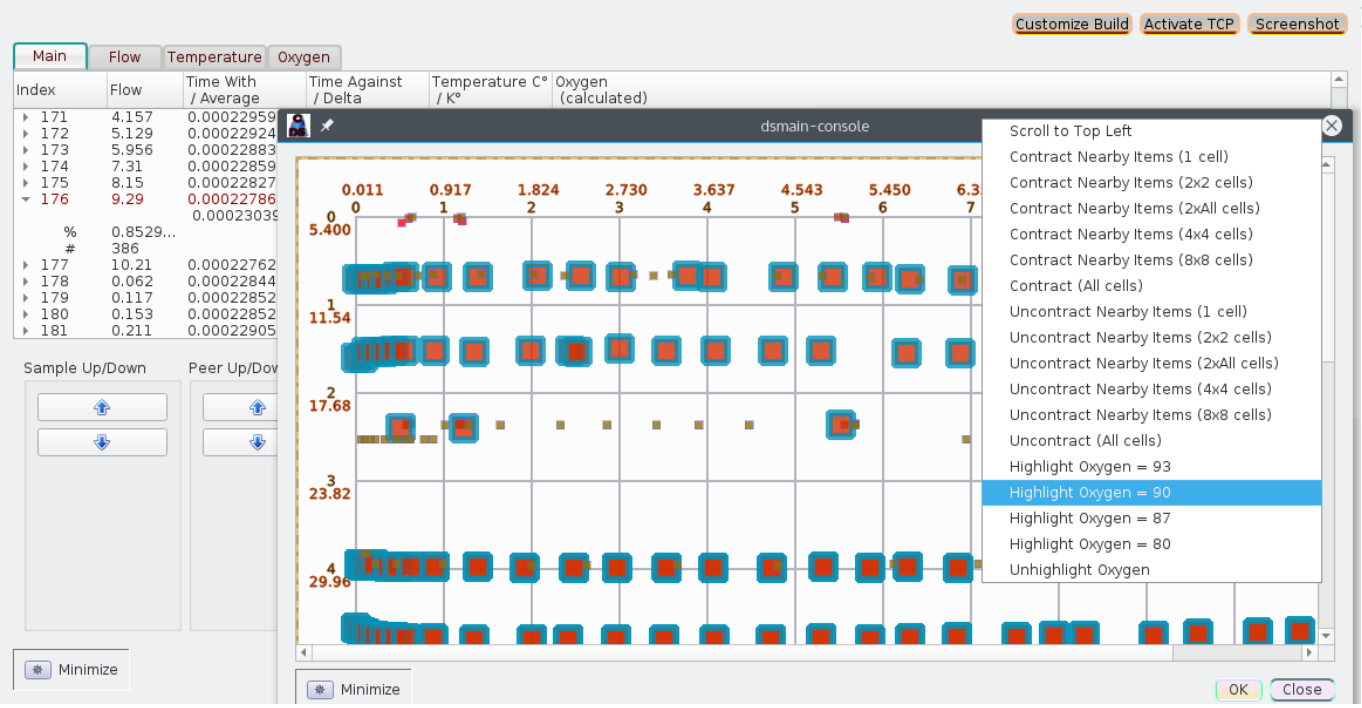
Multi-Application Networks in the Context of Scientific Research Data

Architecturally, the pattern of organization just described — semi-autonomous applications linked together (often by virtue of being common extensions to an overarching "core" software platform) is analogous to the collection of software components that may share access to a data repository or a research-data corpus, include a corpus of medical/diagnostic images. The purpose of research data archives — particularly when they embrace contemporary open-access standards such as FAIR (Findable, Accessible, Interoperable, Reusable[7]) and the Research Object Protocol[8] — is to promote reuse and reproduction of published data and findings, such that multiple subsequent research projects may be based on data originally published to accompany one book or article. As a result, it is expected that numerous projects may overlap in their use of a common underlying data set, which potentially means a diversity of software components implementing a diversity of analytic techniques, each offering a unique perspective on the underlying data. In addition to providing diverse analytic analytic methods, research-oriented software transforms the ecosystem where scientific data and academic books/articles are published and explored. From a reader's point of view, open-access data sets present an interactive, multi-media experience which supplements reading article texts. Indeed, in recent years, publishing houses have embraced the notion (albeit more of a future vision than a present reality) that conventional publications are only one part of a larger package (e.g. a "Research Object Bundle"), which may contain data, computer code, and/or interactive graphics alongside text-based formats such as PDF. At the same time, from a scientist's point of view, open-access research data offers a starting point for their own investigations, or a contretemps with which to reinforce or contrast their own findings.

All of this means that scientists preparaing academic papers are not only finalizing text descriptions

[6] Our proposed enhancements will allow an additional form of plugin using the ReproZip framework; see https://www.reprozip.org/.
[7] See https://www.researchgate.net/publication/331775411_FAIRness_in_Biomedical_Data_Discovery.
[8] See https://pages.semanticscholar.org/coronavirus-research

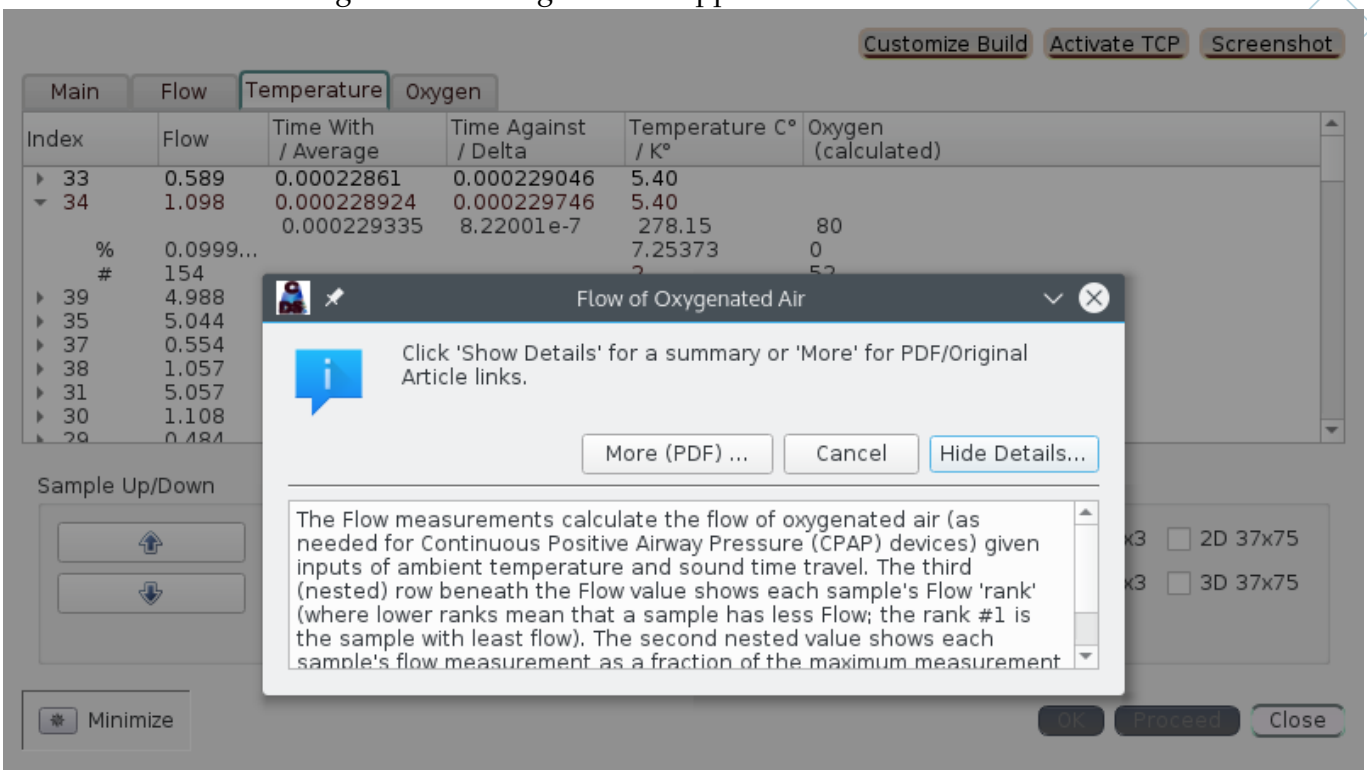
Figure 1: Data Microcitations via Tabular Columns



but also, often, curating data, graphics, or code that demonstrates their work in an interactive, multi-media fashion (such assets are often presented to readers via "supplemental material," "additional files," or "data availability" sections on web pages showing artice texts or abstracts). Scientists can support multi-media exploration of their research by presenting data in file formats used by data-visualization software; they can also assert more fine-grained control over data visualization by implementing custom software. Figure 1 illustrates an example of a cutomized data-set application presenting analysis in the context of biomedical devices (specifically, **CPAP** oxygen-flow monitors) conducted by Dr. James Rodger (Principle Investigator for this project proposal). One benefit of custom data-set software is the possibility of using *microcitations* to connect research data (and the **GUI** components where this data is visualized) to publication texts.[9] Microcitations enable application-level interoperability between document viewers and data-set applications. For instance, individual table columns can be associated with specific scientific concepts or statistical parameters that are described in the article text; Figure 2 shows an example where a user activates a context menu by right-clicking on a column header and is given a list of options, one of which yields to a dialog box that displays a thumbnail explanation of the column data and links to anchors in the article's **PDF** file. This interop between data sets and **PDF** viewers is an example of multi-application networking — both the data-set application and the **PDF** software need to be implemented or extended with the capacity to execute microcitation-based workflows.

[9] Microcitations are independently citeable smaller units of a data set, such as an individual record, a table column, or one table in a multi-table collection; see https://www.thelancet.com/action/showPdf?pii=S1473-3099%2820%2930086-4.

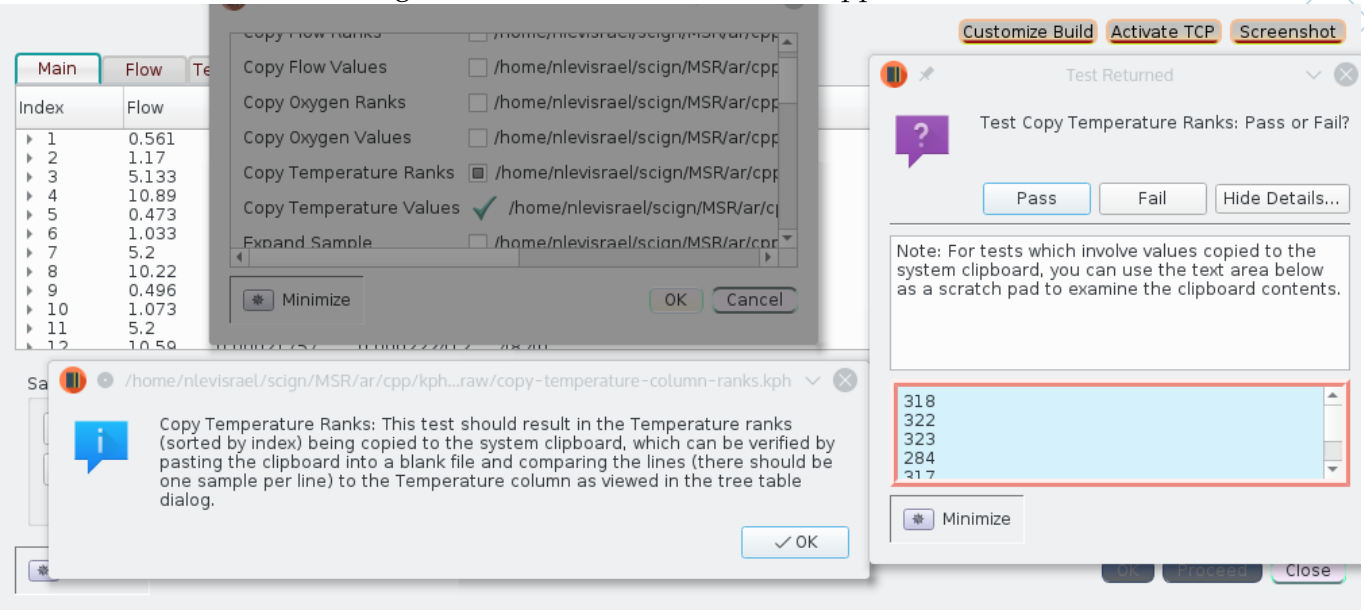
Figure 2: Linking Dataset Applications to Publications



Moreover, data-set applications can provide (through their type system and implementation protocol) one form of structural model and formal elaboration of research theories, methodology, or experimental design. Defining an annotation schema for data sets can potentially be an organic outgrowth of software-development methodology — viz., the engineering steps, such as implementing unit tests, which are essential to deploying a commercial-grade application. This point is illustrated in Figure 3, which shows a GUI-based testing environment for the data set depicted in Figures 1 and 2. For this data set, the context menu actions providing column-specific functionality are also discrete capabilities which can be covered by unit tests, so the set of procedures mapped to the citeable concept correspond with a set of unit-test requirements. In this data set, these procedures are also exposed to scripting engines via the Qt meta-object system. In general, there is often a structural correlation between scripting, unit testing, and microcitation, so that an applications' scripting and testing protocol can serve as the basis for annotation schema and structural models formalizing how a data set is organized.

Implementing a robust research-data software framework involves integrating multiple scientific applications, but also (ideally) extending these applications with features specifically of interest to those conducting or reviewing research using published data sets and/or described in academic literature: for instance, capabilities to download data sets from open-access scientific portals; to parse microcitation formats; and to interoperate with document viewers. Figure 4 shows a case-study of these fearures, where a molecular-visualization application (IQmol) has been extended with a

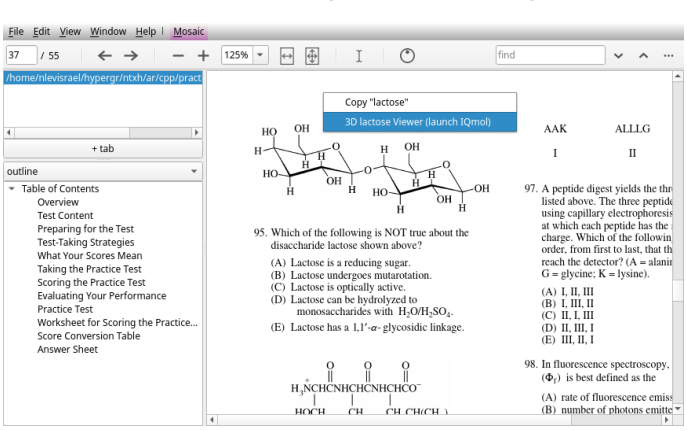
Figure 3: Test Suites for Dataset Applications

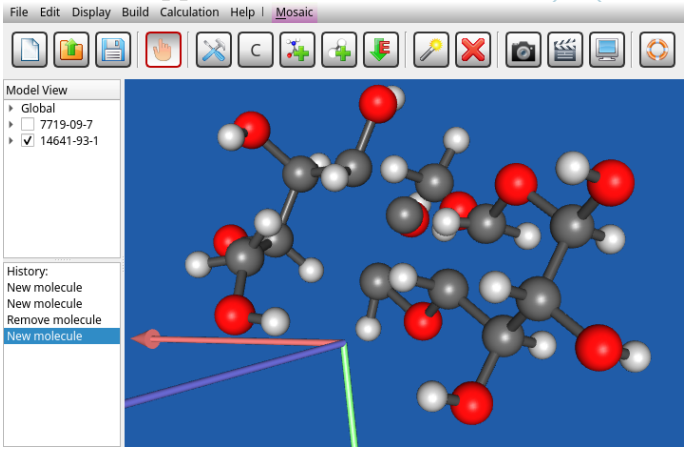


multi-application messaging capacity allowing it to send and receive data from **XPDF** (a **PDF** viewer); in the figures the displayed **PDF** document is a **GRE** practice exam, and **IQMOL** is instructed to present a **3D** graphic visualizing the molecular structure of a compound which is the topic of a test question (the operation to launch **IQMOL** is provided as a context-menu action localized to the page-coordinate area close to the relevant question and answers on the **PDF** viewport. This example illustrates how applications within an integrated research-software environment can complement each other by providing distinct capabilities (such as interactively displaying Protein Data Bank files). A similar example is visible in Figure 5 and Figure 6, which shows a multi-application workflow connecting **3DIMVIEWER** (an example of **3D** radiology software described in the next section) and **MESHLAB** (a general-purpose **3D** visualization and analysis program): **3DIMVIEWER**'s algorithms build a **3D** model from a **2D** image series, and the resulting mesh file is then sent as a data package to **MESHLAB** so that it may be studied in a more fully-featured graphical environment.

This review of data-publishing technology is relevant to radiology and to Patient-Centered Outcomes because it typifies the emerging ecosystem where scientific research and open-access data is being disseminated. Open-access data publishing — especially within the emerging frameworks being advocated and developed by publishers and research institutions themselves — is conducive to a software engineering paradigm that favors the implementation of autonomous software agents which, their autonomy notwithstanding, can interoperate to the degree that they are collectively oriented to a shared data source. Such a multi-application network requires integration not only at the level of data structures, but also at the level of Human-Computer Interaction and inter-application messaging — in the optimal case users can switch between applications based on each components' respective capabilities. In sum, the **CAPTK** architecture — consisting of numerous autonomous, stand-alone, native applications federated into a decentralized but unified platform — is similar logistically to the

Figure 4: Linking PDF Files with Scientific Applications



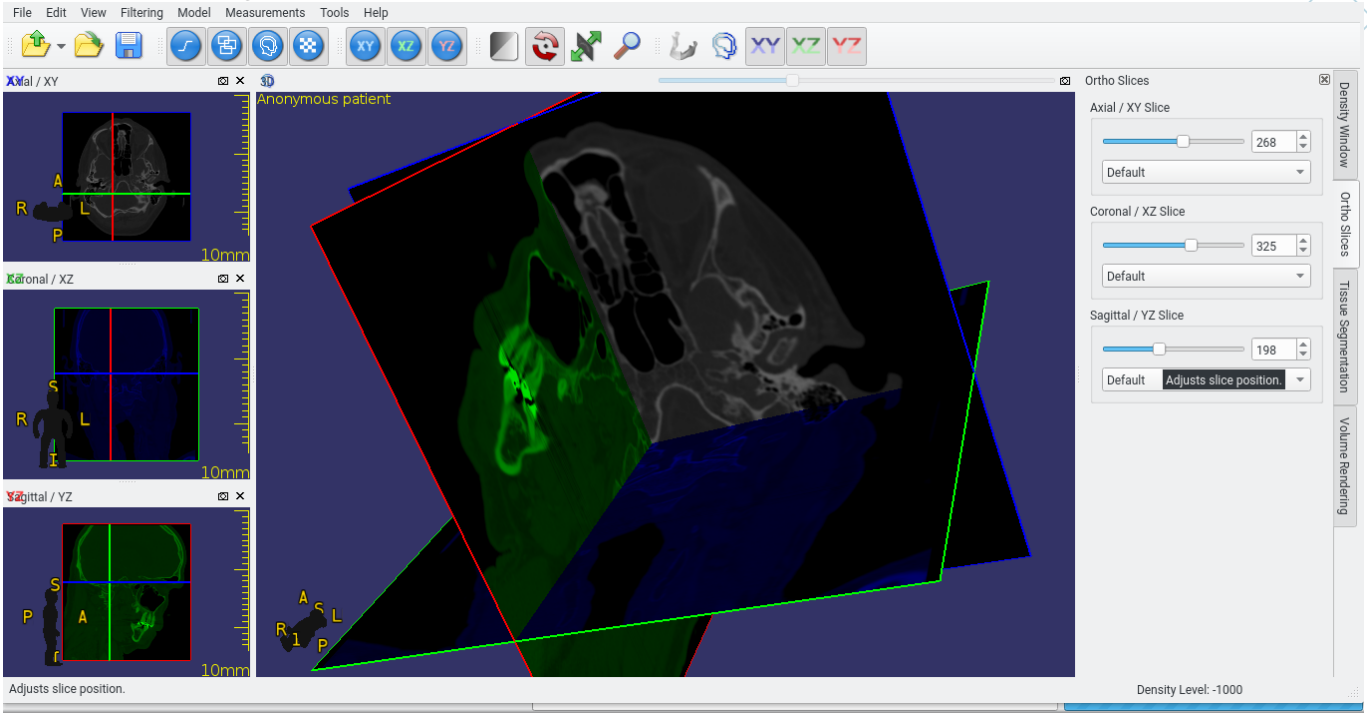


kinds of application networks appropriate for the technology supporting archives of research date (including diagnostic-imaging repositories).

Given this architectural correlation, we propose that the CAPTK architecture can be used as a prototype for implementing multi-application networks such as those applicable to research data repositories. As a starting point for modeling such multi-application networks, we propose to concentrate on software which is implementationally similar to CAPTK within the medical-imaging domain. For example, MEDINRIA (a general-purpose radiology platform) and 3DIMVIEWER (a tool which generates 3D tissue models from 2D image series) are both open-source C++ applications built on top of the QT application development platform, so they occupy essentially the same niche in the software engineering ecosystem as CAPTK. Our proposed CAPTK extensions therefore apply to these related projects, insofar as we can implement plugins extending all three applications with a more rigorous data-integration protocol. Each of these applications likewise lends distinct capabilities to a multi-application radiological platform — whereas CAPTK is focused on image analysis and AI, MEDINRIA functions more as a conventional PACS workstation for manual image-annotation and diagnostic reporting, while 3DIMVIEWER is specifically focused on three-dimensional tissue modeling. A robust data-sharing protocol would ensure that each of these applications can interoperate, allowing users the choice to switch which program they are using depending on the specific analytic tasks they wish to take on for a given session.

Along with these radiology-specific applications, the application network we are hereby describing can moreover include other components commonly used by researchers in a radiological context: software such as XPDF (a PDF viewer), IQMOL (a molecular-visualization program), PARAVIEW (a data-visualization platform), OCTAVE (an open-source Matlab alternative), and QT Creator (a C++ Integrated Development Environment) are all built with the same QT/C++ foundation as the three radiology tools mentioned above, and each of these applications provide capabilities sometimes needed for curating and conducting medical-imaging research. Therefore, by extending each of these applications with a common data-sharing and operational-integration protocol, we can provide scientists with a

Figure 5: Three-Dimensional Tissue Reconstruction via 3DimViewer

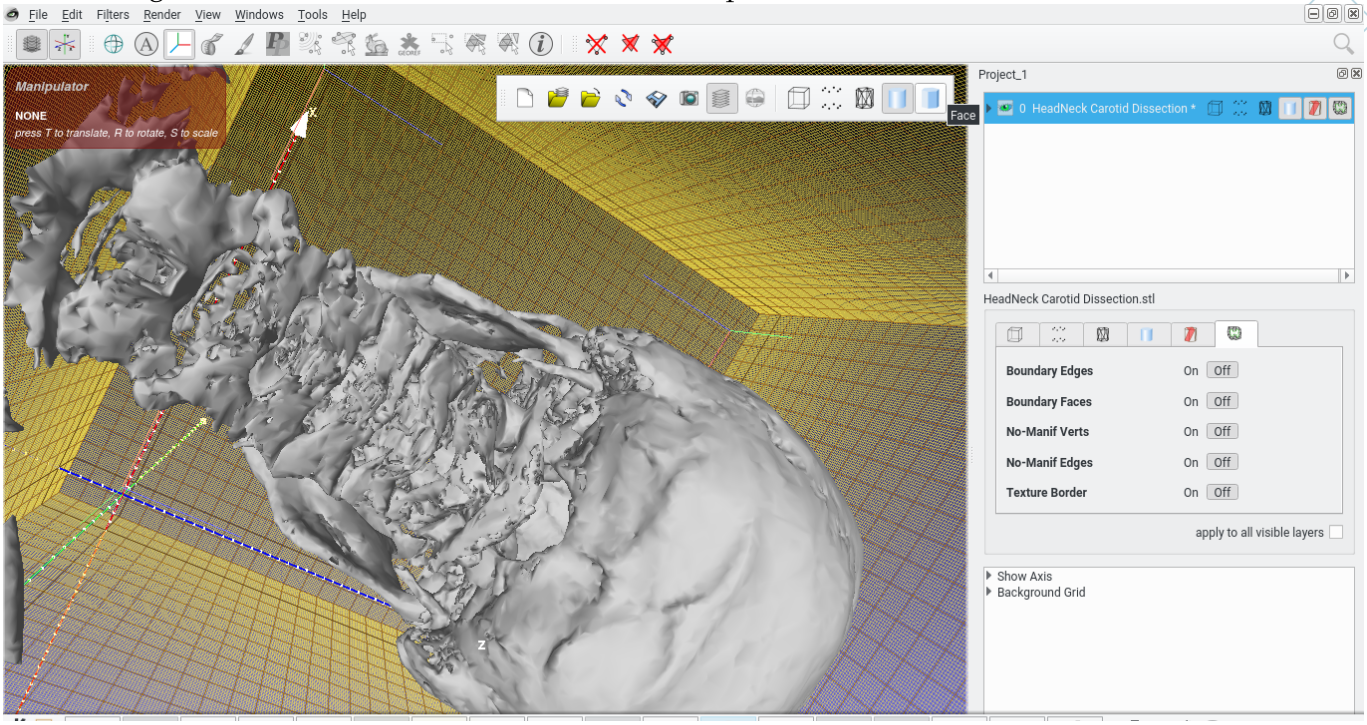


flexible research platform which synthesizes the capabilities of many popular scientific-computing applications. As an initial framework in which to implement this platform, we propose to focus on software tools for accessing Covid-19 research data included in the **RSNA** repository, as well as other Covid-19 archives, such as the **CORD-19** collection of academic publications concerning Covid-19 and SARS-CoV-2.[10]

Initiatives such as Research Objects and **FAIR** advocate for a technological infrastructure characterized by a diverse software ecosystem paired with a open-access research data sets. Scientific data repositories, linked to academic publishing platforms, have been engineered to help scientists locate and re-use data sets which are relevant to their research projects. Although formats such as Research Objects have been standardized over the last decade, there has not been a comparable level of attention given to formalizing how multiple software applications should interoperate when manipulating overlapping data. The Common Workflow Language (**CWL**), which has been explicitly included in the Research Object model[11], documents one layer of inter-application messaging, including the encoding of parameters via command-line arguments; **CAPTK** provides a **C++** implementation of **CWL** and uses it to pass initial data between modules. Serializing larger-scale data structures is of course a generic task of canonical encoding formats such as **JSON**, **XML**, **RDF**, and Protocol Buffers — not to mention text or binary resources serialized directly from runtime objects via, for instance, QTextStream and QDataStream. This means that some level of inter-application

[10] See https://pages.semanticscholar.org/coronavirus-research.
[11] see http://www.researchobject.org/scopes/

Figure 6: The Three-Dimensional Model Exported from 3DimViewer to MeshLab



communications is enabled via **CWL**, and that essentially any computationally tractable data structure can be encoded via formats such as **XML**. These solutions, however, are sub-optimal: **XML** (as well as **JSON** and analogous formats) is limited because it takes additional development effort to compose the code that marshals data between runtime and serial formats. Similarly, although **CWL** can model information passed between applications, it provides only an indirect guide to programmers implementing each applications "operational semantics" — viz., the procedures which must be executed before and after the event wherein data is actually passed between endpoints.

In the context of **CaPTk**, for example, integrating peer modules with the **CaPTk** core application involves more than simply ensuring that these endpoints communicate via a standardized data-serialization format: the plugins must be *registered* with the core application, which affects the core in several areas, including the build/compile process and construction of the main **GUI** window. Modeling the interconnections between semi-autonomous modules, as **CaPTk** demonstrates, therefore require more detail than simply modeling their shared data encodings; it is furthermore necessary to represent all procedural and User Interface requirements in each component that may be affected by the others. Despite the standardization efforts that have been invested in Research Objects and the Common Workflow Language, we contend that this fully detailed protocol for multi-application interop has not yet been rigorously formalized. As part of our project for extending **CaPTk** and then generalizing this extension to scientific research platforms (not necessarily restricted to radiology), we propose to ameliorate this situation by implementing a more rigorous protocol for multi-application networking, which we will call the "Hypergraph Multi-Application Configuration

Language" (**HMCL**). This language is paired with a new data-exchange protocol that we are also implementing, the Hypergraph Exchange Format (**HGXF**), both of which are summarized in the next section.

Our proposed Hypergraph Exchange Format and Hypergraph Multi-Application Configuration Language

To explain the rationale for **HMCL** and **HGXF**, consider the role that data-serialization and Interface Definition languages play in software engineering: programmers need guidelines when implementing procedures wherein one application sends, receives, and/or acts upon data sent or requested by a second application. Standardized representation formats provide specifications that help codewriters serialize data in ways that produce usable resources — representing data via **XML**, for instance, ensures that any application has the theoretical capacity to decode the information (insofar as **XML** libraries exist for every mainstream programming language and environment). Optimally, both endpoints in a cross-application messaging scenario should be able to encode and decode information with as little "biolerplate" code as possible, which is why applications may choose serialization languages which are detailed and expressive, so long as they have guarantees that partner applications can recognize the same format. This is one justification for hypergraph serialization, insofar as hypergraphs are a very flexible representation that can accommodate almost any data structure that needs to be encoded, often with less translational effort than marshaling data into formats such as **XML** and **JSON**.

Hypergraphs provide a generalization of graph-like and/or hierarchical data structures in multiple contexts: in markup languages, for example, hypergraphs allow for concurrent or overlapping node-structures, yielding formats which are more expressive than **XML**. In the Semantic Web, hypergraphs are used as a generalization of **RDF**, allowing for nested levels of structure and for data localization. Within image analysis, hypergraphs are often employed to represent feature vectors and/or geometric relations between image segments. Hypergraphs have also been utilized in the description of multi-part biological or biomedical processes: protein complexes, metabolic reactions, genomic sequences, and so forth. Considering the range of specialized cases for hypergraph representations — as well as how hypergraph structures are suitable for simpler data that could also be represented via **XML** or **RDF** — it is reasonable to select hypergraphs as a generic data-encoding format. We propose a flexible hypergraph model which can accommodate various hypergraph categories (e.g., both directed and undirected) and can represent supplemental information sometimes introduced into hypergraph models, such as probabilities (inserting measures of uncertainty or fuzziness into relations) and "roles" (characterizing multi-part relations in terms of the situational status of their constituents).[12] The general category of hypergraphs encompasses a range of structures implemented by various

[12] The concept of roles is highlighted by Grakn.ai: see https://dev.grakn.ai/docs/schema/concepts.

hypergraph libraries and database engines. Although several graph-serialization formats currently exist, we believe a new Hypergraph Exhange Format is warranted which is intrinsically designed to model the full range of hypergraph structures developed in the scientific literature as well as the scientific-computing community.

This discussion illustrates the rationale for designing our **HGXF** format. As mentioned above, this format is intended to be paired with a Hypergraph Multi-Application Configuration Language (**HMCL**). The purpose of **HMCL** is to represent multi-application networking protocols with greater procedural detail and specificity than provided by traditional workflow or Interface Definition languages. Within the scope of multi-application workflows, we have, in prior paragraphs, briefly outlined the protocols adopted by **CaPTk**. A more rigorous review of multi-application networking can be obtained by considering analogous protocols connecting semi-autonomous software agents in other contexts: **ParaView** extensions, **Qt** Creator Plugins, **Octave** scripts, and **ROOT** modules, for example (**ROOT** referring to the physics platform developed at **CERN**) — restricting attention to the **Qt**/**C++** ecosystem. Further afield, a similar sense of semi-autonomy can be found in hybrid **GUI** and scripting platforms such as Jupyter (in the Python context) and Seco (a Java-based "notebook" framework built on top of HyperGraphDB, where *notebook* in this environment refers to interactive code development and execution containers such as Jupyter and RStudio). These platforms are familiar to data visualization and machine-learning engineering (where a script may analyze data and a **GUI** component follow up by presenting a chart or dataplot summarizing the analysis), as well as other quantitative domains outside the natural scientists (in financial services, for example, scripting formats such as **MQ4** execute investment strategies while associated **GUI** components present finance-related visual objects, such as candlestick charts.) Similar workflow models have been developed in theoretically-informed frameworks such as **VISSION**, which is paired with programming constructs such as "metaclasses" and "dataflow interfaces".[13] The common element in all of these systems is the presence of a central **GUI** application whose capabilities are enhanced by customizable extensions with their own analytic and/or **GUI** features: these extensions are neither fully autonomous applications nor simple scripts that may automate certain capabilities of the main application but do not fundamentally extend its functionality. This intermediate position — neither wholly autonomous nor functionally subservient — is expressed by the idea of *semi-autonomous* software agents.

Rigorous models of application networks among semi-autonomous components acquire an extra level of complexity precisely because of this intermediate status: protocol definitions have to specify both the functional interdependence and the operational autonomy of different parts of the application network. We propose to address this complexity by adopting a hypergraph-based paradigm for procedural and type modeling, from which derives our proposed **HMCL** format.[14] The goal

---

[13] See https://pdfs.semanticscholar.org/1ad7/c459dc4f89f87719af1d7a6f30e6f58dff17.pdf; note that this is a different project than the ViSion Ontology referenced earlier.

[14] A preliminary characterization of hypergraph-based type systems (and their corresponding representation of detailed procedural signatures and requirements) can be found in Nathaniel Christen's chapter (chapter 3) in *Advances in Ubiquitous Computing: Cyber-Physical Systems, Smart Cities and Ecological Monitoring*, edited by Dr. Neustein (see

of HMCL is to define protocols for inter-application messaging by focusing on procedures enacted by both applications before and after each stage in the communication. This is not (in general) a "Remote Method Invocation" or "Simple Object Access Protocol" where one application explicitly initiates a specific procedure for the second to perform; instead, the chain of procedure calls (which we call the multi-application *operational semantics*) is more indirect, with *requests* and *responses* that may be mapped to different procedure-sequences in different contexts. Nevertheless, although one application does not need detailed knowledge of the other's internal procedure signatures (which would break encapsulation), a rigorous messaging protocol can be developed by specifying requirements on the relevant procedures implemented by each application. Developers can then explicitly state that a given set of procedures implements a given HMCL protocol (the multi-application documentation thereby has more detailed information about application-specific procedures than each application has vis-à-vis its peers). The functional interdependence between applications can accordingly be modeled by defining protocols which must be satisfied by procedure-sets internal to each end-point — the relevant information from an integrative standpoint is not the actual procedures involved, but confirmation that the relevant procedure sets adhere to the relevant multi-procedural protocol.

Reviewing the source code and documentation for CaPTk confirms that multi-application messaging along these lines is implicitly adopted by CaPTk, but the protocols involved have not been formalized with the level of rigor we contend is possible via HMCL. As the central element in our extension to CaPTk, we propose therefore to formalize the CaPTk protocol using HMCL, which among other benefits will allow plugins to be introduced into CaPTk via configuration files rather than through the manual process described in the CaPTk literature.[15] At a practical level, we believe these enhancements to CaPTk will make it a little easier for CaPTk to be used in a context where researchers wish to analyze clinical outcomes in conjunction with image data within the scope of statistical, Machine Learning, or AI methods. Moreover, they will allow CaPTk to be integrated with other radiological software (such as MedInria and 3DimViewer) and, more generally, with a suite of scientific applications, arriving at a multi-purpose research platform optimized for the curation and re-use of open-access research data. At a more theoretical level, we also believe that the formats we have described here — specifically HGXF and HMCL — can be useful in a wide range of scientific computing and software engineering contexts, particularly in conjunction with "reference implementations" that we will provide in the context of CaPTk.

---

https://www.elsevier.com/books/advances-in-ubiquitous-computing/neustein/978-0-12-816801-1).

[15] See for example https://www.med.upenn.edu/cbica/assets/user-content/images/captk/2018_ISBI_CaPTk.0404.Part2.pdf, particularly the material starting on the 30th slide of that presentation.

Hypergraph Ontologies, and a concrete Hypergraph Ontology for radiological outcomes

As explained above, the goal of **HGXF** is to provide a general-purpose hypergraph representation format, suitable for all hypergraph data structures as well as any structures which are categorically subsumed by hypergraphs (such as **RDF** graphs). Consequently, **HGXF** is designed in part by examining existing Hypergraph Database software (such as HyperGraphDB, AllegroGraph, and Graken.ai) and runtime hypergraph libraries, as well as academic literature where hypergraph analyses have been used for (e.g.) image-segmentation and Machine Learning algorithms. These resources provide an overview of the range of data structures which need to be encoded by a general-purpose language such as **HGXF**. The design of **HGXF** has also been influenced by the Semantic Web, insofar as hypergraphs are a generalization of the directed, labeled graphs that are the building-blocks of the Semantic Web. At the same time, hypergraphs — along with structures that can be modeled via hypergraphs, such as Conceptual Spaces — are an improvement on Semantic Web data formats and schema, addressing the limitations of a paradigm devoted to modeling complex information via first-order logic and non-nested graphs, with no notion of scoping or locality.[16]

At the same time, a lot of effort has been extended building technologies to integrate heterogenous data spaces via the Resource Description Framework (**RDF**) and **RDF** ontologies; it would be irresponsible to ignore these contributions. In radiology, for example, attempts to better incorporate clinical and outcomes data are centered on ontologies such as **RadLex**, **VISION**, and **SEDI**; insofar as these are (or have potential to become) canonical reporting standards, Patient-Centered research should promote rather than critique these initiatives. The important consideration, then, is how to employ hypergraphs as an extension to the Semantic Web when warranted while preserving the virtues (and interoperating with) conventional **RDF** ontologies.

The idea that hypergraphs *extend* but do not *replace* **RDF** and the Semantic Web implies that hypergraph schema are extensions of (but not substitutes for) **RDF** ontologies — which in turn yields the notion of *hypergraph ontologies*. In a conventional **RDF** ontology, metadata is primarily associated with graph nodes and edges. In particular, nodes are referenced to Uniform Reference Identifiers (**URI**s), such as web addresses, and edges are labeled with concepts formally defined in one or more ontologies. Concepts which are used to annotate graph-edges, and which are given a fixed meaning in some controled vocabulary, are often called "properties." One special "is-a" property is often used to connect nodes with concepts that classify entities into one of many categories defined in an ontology, often called "classes." As such, most **RDF** ontologies are primarily composed of *classes* and *properties*, each assigned a unique label. The purpose of metadata for a given graph is then

[16] These are familiar critiqued, but laid out with particular thoroughness by the Conceptual Space community: see http://idwebhost-202-147.ethz.ch/Publications/RefConferences/ICSC_2009_AdamsRaubal_Camera-FINAL.pdf and https://pdfs.semanticscholar.org/1d58/faa5cb23efb7394aeb3dfe688edd99797a91.pdf.

to link nodes to classes (for example, specifying that one node represents a clinical trial and the second represents a patient), and furthermore link edges to properties (for example, specifying that a patient-node is connected to a trial-node in that the patient is *enrolled in* the trial).

Hypergraph ontologies are similar to conventional **RDF** ontologies in that they likewise provide constraints and metadata for graphs. However, hypergraph ontologies are more complex because hypergraphs are likewise more complex than ordinary graphs. In particular, hypergraphs have different layers of structure: whereas **RDF** nodes are intended to represent a single concept or value (such as a number, date, personal name, or **URL**), a *hypernode*, within a graph, typically encompasses multiple pieces of information inside it (often called *hyponodes*, *projections*, *inner elements*, *roles*, or just *nodes*).[17] In general, when analyzing hypergraphs it is necessary to distinguish at least two "tiers" of nodes, *hypernodes* and *hyponodes*, such that hyponodes are contained within hypernodes. As a result, hypergraph ontologies need a corresponding distinction for node and edge annotations: insofar as nodes are categorized via classes, and edges via properties, it is neccesary to stipulate whether these classifications apply to hypernodes, hyponodes, or some combination of the two.

A further complication (contributing to the complexity of hypergraph ontologies compared to **RDF** ontologies) arises because, even though hypergraphs represent nested or hierarchical structures, these hierarchies are often partial or overlapping. For example, a patient is *part of* a clinical trial, but a patient is also included in other collections as well; for instance, a patient may be enrolled in a specific health plan (for insurance coverage). One technique for modeling overlapping hierarchical data via hypergraphs is to employ "proxies," which are digital identifiers encoding a multi-faceted concept into a single value that can be part of a hypernode (proxies are similar to "foreign keys" in **SQL**). Therefore, each patient, represented by its own hypernode, has an identifier which can be a proxy-value for the patient; for example, a value assigned to a hyponode becomes included in the hypernode encoding the list of patients enrolled in a clinical trial, or in the hypernode encoding the list of patients enrolled in a specific health plan. Hypernodes can then be linked to other hypernodes by virtue of proxies (e.g., the trial-to-patient connection), and also by virtue of overlap (e.g., the set of all patients both enrolled in a given clinical trial *and* enrolled in a given health plan).

In sum, compared to **RDF** — where there is one single sort of node-to-node relationship, based on whether or not an edge exists betweel nodes and how this edge is labeled — hypergraphs are more flexible/expressive because they have multiple genres of node-to-node relationships: the relation between hypernodes and their inner hypnodes; between hypernodes and one another; between hyponodes in different hypernodes; and variations on each of these relation-types wherein relations are defined indirectly through proxies. Moreover, in addition to hypernodes and hyponodes, hyper-

---

[17] The term "projections" is used by HyperGraphDB (see http://www.hypergraphdb.org/?project=hypergraphdb&page=RefCustomTypes); "roles" is used by Grakn.ai (see https://dev.grakn.ai/docs/schema/concepts); "inner entity" is used by the biointelligence project (see https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3555311/), where the corresponding notion of "external entity" refers to what in other contexts might be called other hypernodes linked to a given hypernode via hyperedges.

graphs afford additional levels of detail, such as *frames*, *channels*, and *axiations*. All of these details provide different "sites" where hypergraph annotations and metadata may be defined.[18]

An additional distinction within the Semantic Web is the contrast between *reference ontologies* and *application ontologies*. In general, *reference ontologies* are general-purpose schema intended to establish conventions shared by many different applications, to ensure that a large collection of data-producing software in a given domain is interoperable. By contrast, *application ontologies* are narrower in scope because they are more tightly integrated into applications that directly send and receive data. Ontologies of either variety are used by software to interoperate with other software: so long as two applications are using the same ontologies, it is possible to ensure that one application can understand the data produced by a second, and vice-versa. However, such inter-operability is only a potential; it is the responsibility of programmers to actually implement code which produces and/or consumes data that conforms to the relevant ontology specifications. In general, application ontologies are structured in such a way that these concrete implementations are more straightforward to produce, compared with reference ontologies. Reference ontologies offer little guidance to developers vis-à-vis how to directly support the ontology within application code. Conversely, application ontologies more rigorously describe the data structures which applications must implement in order to properly manipulate data that is structured according to the specifications of the ontology.

Within data mining and image analysis, hypergraph models are used in different ways for different algorithms. In the context of Covid-19 radiology, https://arxiv.org/pdf/2005.04043.pdf describes an algorithm for assessing the probability of SARS-CoV-2 infection from chest **CT** scans, where hypernodes represent high-dimensional vectors (191 dimensions overall) and hyperedges represent k-nearest-neighbors; here each hypernode represents an entire image, mapped to a 191-dimensional feature-vector. In contrast, other image-analysis methods use hypernodes to designate smaller segments *within* the image, where hyperedges designate geometric adjacency and/or feature-space similarities. Whatever the algorithm, hypergraph analyses would employ a hypergraph library to store preliminary data for analysis and/or for serialization within a data set. One benefit of a Hypergraph Application Ontology is therefore that these data structures used internally to implement analytic methods can be directly expressed within the ontology, whereas **RDF** ontologies can only model hypergraphs indirectly.

Although it is theoretically possible to encode data directly via **RDF** graphs, it is far more common for applications to employ tabular and/or hierarchical formats such as spreadsheets, Protocol Buffers, **XML**, or **JSON**. As a result, the role of ontologies for constraining data structures (so that they adhere to common standards) is indirect. It is useful to remember that ontologies are, at their most basic level, Controled Vocabularies; as such, ontology constraints often amount to stipulating

[18] This means that formats for describing hypergraph ontologies have to be more expressive than **RDF** ontologies, because **RDF** ontologies need only to classify metadata as node-annotations or edge-annotations; by contrast, hypergraph ontologies need to distribute annotations among multiple sites of graph structure.

a set of acceptable terms for a data value, column header, or annotation. For a trivial example, our calendar recognizes 12 month names and 7 day names, which constrain the set of values permissible for "month" and "day" within a calendar date. These terms are so commonplace that a "date ontology" is unnecessary, but in scientific or technical domains it becomes necessary to define vocabularies of allowable names or labels for specific data fields that representing some scientific value or measurement. For instance, the Ontology of Vaccine Adverse Events (see https://jbiomedsem.biomedcentral.com/articles/10.1186/2041-1480-4-40) provides a nomenclature for use in Adverse Events Reporting, so that researchers or clinicians can describe symptoms via canonically recognized terms rather than through informal text descriptions. In general, ontologies constrain data sets by stipulating that particular individual values within the overall data collection have names or descriptions whose associated set of possible values is prescribed *a priori* by the applicable ontology. However, the relationship between ontologies and concrete data sets must itself be documented, which is where application ontologies can become relevant — application ontologies provide a bridge between reference ontologies and the applications which use them (along with the data generated and shared by those applications).

In order to preserve the benefits of **RDF** ontologies — while also addressing those lacunae which make the Semantic Web "not [really] semantic," — hypergraph ontologies need to model constraint schema on hypergraph constructions (which have significantly more parameters of structuration than **RDF** graphs) while also connecting these schemas to the Controled Vocabularies and logical axioms of Semantic Web (particularly **OWL**) ontologies. There are as such several areas of detail within hypergraphs where links to **RDF** ontologies may be drawn, which are outlined here:[19]

Hypernodes' Cocyclic Type Structure One of the central principles of hypergraph data modeling is the use of *hypernode types* to specify what sort of information is necessarily associated with a hypernode. In particular, a hypernode encompasses multiple hyponodes, each with their own type. These hyponodes represent information in some sense "contained within" or "tightly connected to" a hypernode (whereas data less canonically associated with each hypernode would, in general, be asserted via hyperedges rather than via hypernode/hyponode containment. In order to ensure that hypergraphs are predictably organized, hypernodes cannot have arbitrary collections of hyponodes, but must instead be aggregates of hyponodes which are assembled according to a schematic pattern, defined in terms of hyponode types. For maximum generality, a hypergraph type system should allow hyponode-type patterns to be as flexible as possible without introducing a need for metadata asserted at the level of individual hypernodes rather than hypernode types; this motivates the idea of a "cocyclic" type system which is minimally constrained (but not unconstrained).[20] When translating **RDF** ontologies to hypergraph schema, then, one consideration is

---

[19] A full explanation of these concepts and terminology depends on an in-depth treatment of hypergraph type theory, which is outside the scope of this proposal.

[20] A pattern of hyponode types can be called "cocyclic" if the type-sequence includes a (possibly empty) tuple of types with no requisite pattern (called the "precycle") followed by a repeatable type-tuple. Cocyclically typed hypernodes

whether edge-requirements are sufficiently ubiquitous in some context (e.g. with respect to some **rdf:class**) that these edges should be translated to hypernode/hyponode inclusions, and then to define a pattern of hyponode types for the corresponding hypernode type.

Roles, Projections, and Dimensional Annotations A hypernode type provides a schema defining a sequence of hyponode types; it is sometimes said that the hypernode "projects onto" that space of hyponode types. This projection is minimally characterized by hyponode types, but some hypergraph systems allow the projection to be *annotated*, introducing additional metadata that constrain (or augment the expressive power of) the enclosing hypergraph. Annotations can define scales/units/levels of measurement, probability distributions, situational roles, and other details lending semantic grounding to the data-field encapsulated by a hyponode. This metadata can then be a vehicle for translating **RDF** class constraints to hypergraph schema.

Semantic Nominal Dimensions The most direct translation of Controled Vocabularies to a hypergraph context is often that of constraining the space of variation for one specific project to a set of allowable terms. In the typical case, a hyponode type encapsulates a nominal set of values (i.e., an enumeration), so any hypernode including that type as one of its projects is constrained by the labels registered in the vocabulary (a related formulation replaces non-hierarchical vocabularies with *taxonomies*, where some labels are treated as more or less granular variants of others).

Dimension Aggregates, Domains, and Conceptual Spaces Conceptual Space Theory can be modeled in the hypergraph context by noting that hypnode projections are sometimes interdependent: dimensions tend to aggregate into semantically related groups (like *latitude* and *longitude* as geographic markers). In Conceptual Space models, then, projections are split into two levels — *dimensions* and *domains* — while other dimensional-analytic constructions (such as scales and units of measurement) are carried over.[21] Conceptual Space Theory then introduces concepts of fuzzy logic or "convexivity" (according to different metrics) to simulate patterns in human conceptualization.[22]

Probabilistic, Temporal, and Overlapping Hypergraphs Other forms of metadata constrained via ontologies can be expressed in terms of annotations defining weights or probabilities on hypernodes and/or hyperedges. One example is the juxtaposition of alternative markup hierarchies, in the context of hypergraph representations for Concurrent Markup languages such as . Numeric edge-annotation can represent weights (e.g., provide measures of the degree of uncertainty in the edge's relation actually obtaining), but constructions similar to weights have other sorts of appli-

---

therefore represent expandable data structures such as lists, stacks, queues, deques, and dictionaries. A typical hypernode type may indirectly include multiple collections-types via proxies.

[21] See https://pdfs.semanticscholar.org/521a/cbab9658df27acd9f40bba2b9445f75d681c.pdf, https://arxiv.org/pdf/1703.08314.pdf, or http://lup.lub.lu.se/record/1775234.

[22] A good review is provided by the publications and code archived at https://github.com/lbechberger/ConceptualSpaces.

cations. For instance, edge-annotations can be measures of time-spans, allowing hypergraphs to describe "entity-event models."[23] These models are particularly inportant for patient-centered data via their use in analyzing clinical outcomes by aggregating data according to patient-centered reviews extracted from data.[24]

Proxies, Inverted Proxies, and Double-Inverted-Proxy Constructions As described earlier, hypernodes can assert "containment" of other hypernodes by containing a *hyponode* which *proxies* the second hypernode. An *inverted proxy* connection is therefore the mirror-image of this assertion (which may or may not be formally regognized by a hypergraph). A *double-inverted-proxy* connection is accordingly the relation obtaining between two hypernodes which are both proxied by one third hypernode (using the earlier example of proxies, the fact that two patients are enrolled in the same clinical trial). Many graph connections identified in a Semantic Web context (e.g., by **SPARQL** queries) are likely to translated to double-inverted proxies in a hypergraph context.

In general, these hypergraph construction represent sites for asserting constraints that (for **RDF** ontologies) would be defined on classes or properties; they are therefore the framework for translating **RDF** ontologies to hypergraphs. Such a translation mechanism allows existing ontologies — which may play a valuable role in specifying protocols for workflows and data-sharing between software components — to be reused in a hypergraph modeling environment.

As illustrated by **CAPTK**, multi-application workflows are characterized both by the data which is transferred between applications and by the operations which connect the two applications — that is, the procedures enacted by each application when they become operationally linked. As a preliminary model, we can identify two stages of operational connection between an already-running application (which may be called the *primary* component) and a second application launched by the primary (which may be called the *peer* component). The first stage occurs when the primary component launches the peer component, and is characterized by two operational sequences: procedures enacted by the primary prior to this launch, and procedures enacted by the peer subsequent to the launch. A second stage occurs when the peer component has completed its actions, and sends data back to the primary component, which again involves two operational sequences: procedures enacted by the peer prior to the transfer, and procedures enacted by the primary subsequent to the transfer. Fully describing the procedural workflow therefore entails specifying four operational sequences: primary, then peer, during the launch stage; and peer, then primary, during the transfer stage. A schema describing the operations performed during these four sequences can be called a *procedural ontology*.

Consequently, rigorous models of multi-application networks should *synthsize* information about data structures (the type of information shared between application-points) with information about

---

[23] See https://allegrograph.com/consulting/entity-event-knowledge-graphs/.
[24] See http://exploreclg.montefiore.org/upload/training-materials/The%20Cohort%20ParadigmV30.pdf.

procedural workflows (describing operational sequences prior to the launch and transfer stages of a multi-application linkage). The synthesis of this structural and procedural information can be called a *procedural application ontology*. Insofar as the new **HMCL** language defines the operational semantics of multi-application workflows, **HMCL** can be seen as a format for asserting and integrating procedural application ontologies (according to this definition).

Insofar as **PACS** and **DICOM** systems need to be paired with applications in the other categories (such as image and data-set analysis), it is important to develop radiological Application Ontologies, alongside the canonical Reference Ontologies associated with diagnostic imaging. As such, starting with the Covid-19 context, but designed for more general applications as well, our proposal includes the development of a novel radiology-focused *application* ontology, specifically the "Hypergraph Ontology for Diagnostic Imaging, Clinical Outcomes, and Data Mining" (**h-DICOM**). The goal of **h-DICOM** is to connect **RDF**-based radiological ontologies with software applications designed to manipulate radiological (and corresponding treatment and outcomes) data. In addition, **h-DICOM** will formalize the data-sharing protocols implicitly adopted by software aggregates such as **CaPTk** (this part of the ontology will be expressed via **HMCL**).

Conclusion

In sum, then, the following represents contributions that we propose for inclusion into patient-centered outcomes research in the Covid-19 context, as prioritized by the current funding opportunity: first, we will implement an extended radiological software platform which both utilizes and improves upon the extension mechanism present in **CaPTk**, and then integrates other radiological software as well, such as **MEDINRIA**. We will develop extensions that allow researchers to cross-reference radiological and clinical/outcomes data, by introducing more sophisticated data models and protocols for sharing patient-centered information among points in a multi-application network, such as those present between **CaPTk** and its extensions, or (given our expanded platform) between **CaPTk** and other radiological applications, such as **MEDINRIA**. Second, we will implement a multi-faceted research platform oriented toward open-access scientific data sets, particularly the **AI** Image Repository for Covid-19 currently being curated by **RSNA** (and scheduled to be published within the time-frame of this project-proposal). Third, we will formalize and provide reference implementations for a new data serialization format (**HGXF**) and a new interface and workflow description language (**HMCL**) which are applicable to a wide range of scientific domains; we will ensure that these languages are particularly optimized for representing patient-outcomes data and for configuring inter-application messaging in contexts where special protocols have to be observed so as to curate medical information in ways that prioritize patient-centered outcomes.

The implementations hereby proposed as contributions to the PCORnet initiatives coincide with software being developed for two academic volumes. This includes a forthcoming Elsevier volume

titled *Cross-Disciplinary Data Integration and Conceptual Space Models for Covid-19*, co-authored by Dr. Amy Neustein and Nathaniel Christen (Dr. Neustein is the Administrative Officer for this proposal and Nathaniel Christen is the developer who has implemented prototypes for the **HGXF** and **HMCL** languages described here), and a volume focused on medical image processing, titled *Medical Image Processing and Machine Vision*, by Dr. Amita Nandal, who is a contributor to the currently proposed project. The Elsevier Covid-19 volume will be paired with an open-access Covid-19 data repository, aggregating published research relevant to Covid-19 and SARS-CoV-2 from different disciplinary perspectives, including vaccinology, biomechanics, genomics, radiology, and epidemiology. More information about this archive, called the "Cross-Disciplinary Repository for Covid-19 Research" (**CR2**), is available on the github page that will be used for republishing the relevant data (see https://github.com/Mosaic-DigammaDB/CRCR).