



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 cm³” 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.

¹See <https://bioportal.bioontology.org/ontologies/SEDI>.

²See https://epos.myesr.org/esr/viewing/index.php?module=viewing_poster&task=&pi=155548.

³See <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5119276/>

⁴See <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

⁵See <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, Reusable⁷) and the Research Object Protocol⁸ — 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 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 preparing academic papers are not only finalizing text descriptions

⁶Our proposed enhancements will allow an additional form of plugin using the ReproZip framework; see <https://www.reprozip.org/>.

⁷See https://www.researchgate.net/publication/331775411_FAIRness_in_Biomedical_Data_Discovery.

⁸See <https://pages.semanticscholar.org/coronavirus-research>



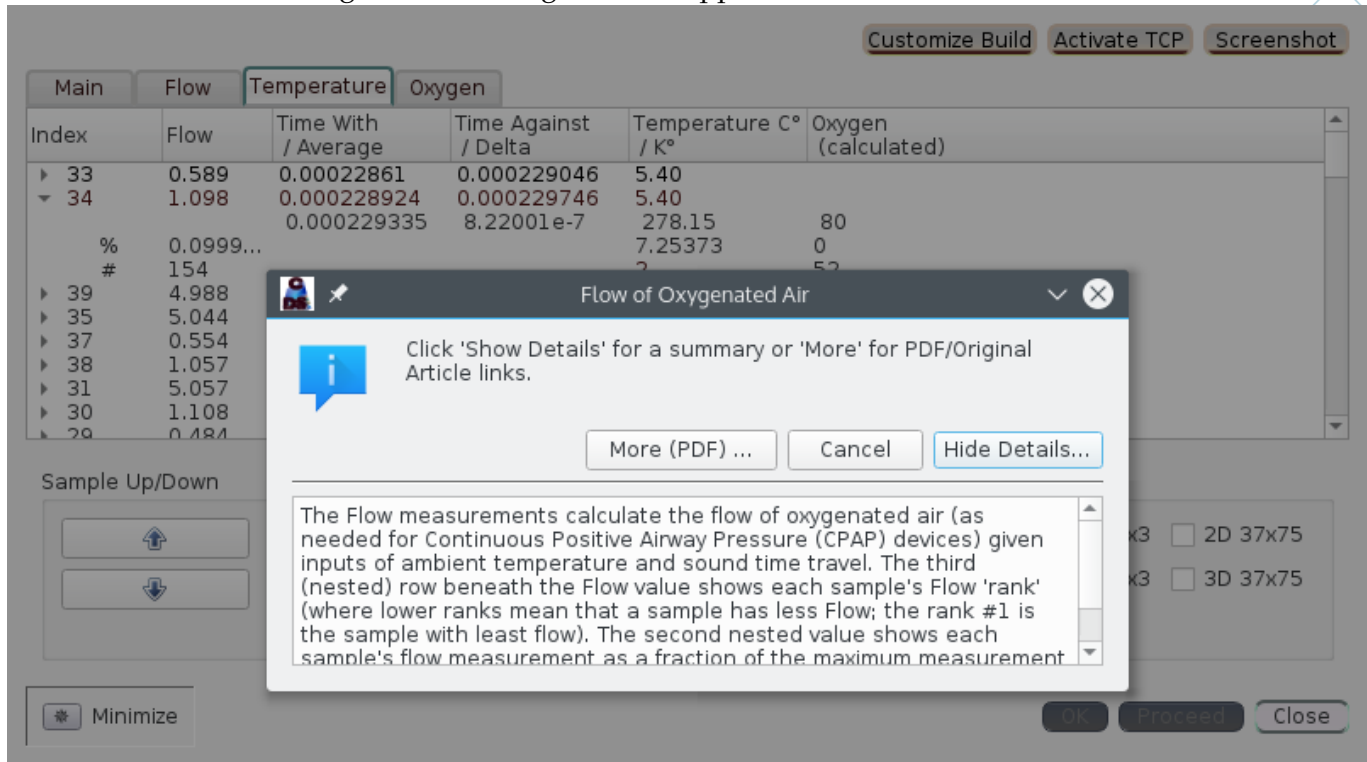
Figure 1: Data Microcitations via Tabular Columns



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 article 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 customized 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>.

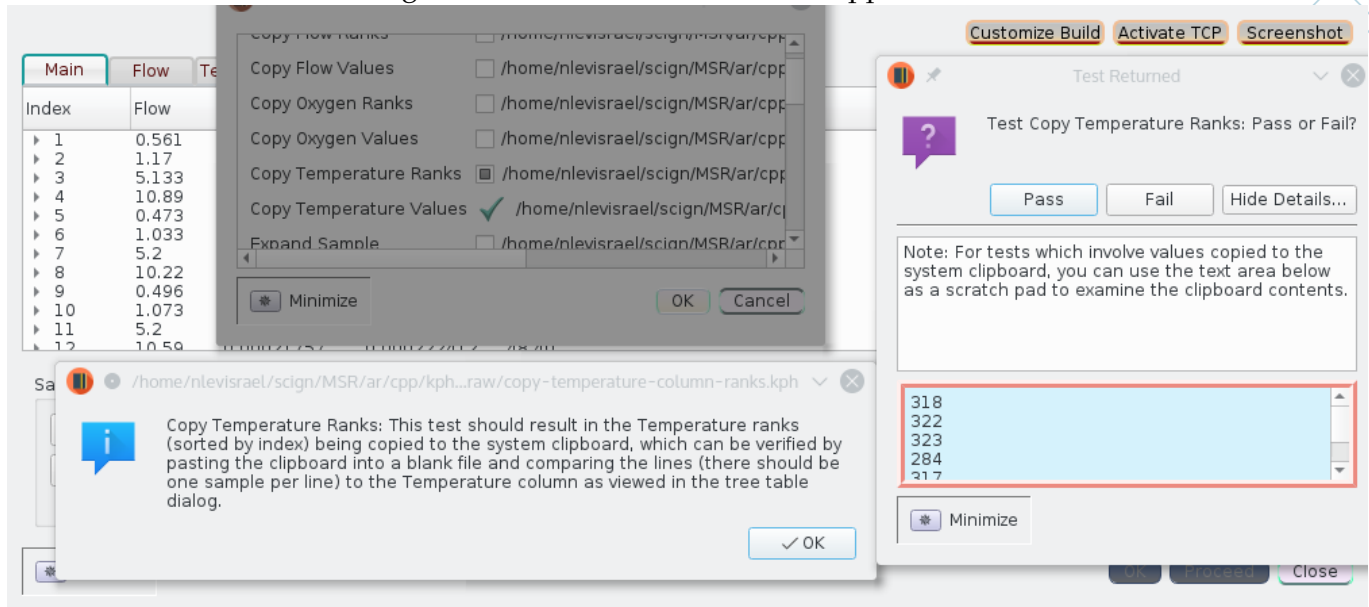
Figure 2: Linking Dataset Applications to Publications



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.

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

Figure 3: Test Suites for Dataset Applications



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 kinds of application networks appropriate for the technology supporting archives of research data (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 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 model¹¹, 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 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

¹⁰See <https://pages.semanticscholar.org/coronavirus-research>.

¹¹see <http://www.researchobject.org/scopes/>

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 hypergraph libraries and database engines. Although several graph-serialization formats currently exist, we believe a new Hypergraph Exchange 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

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

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



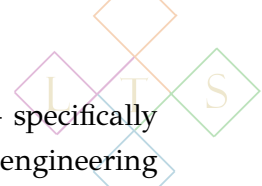
nor functionally subservient — is expressed by the idea of *semi-autonomous* software agents.

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 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.

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 <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.



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**.

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/CR2>).

