MOSAIC

(Multiparadigm Ontologies for Scientific, Academic, and Technical Publications)

SDK and Software Engineering Tools

The MOSAIC Portal is a cloud application for hosting scientific, academic, and technical publications. Together with a collection of server-side and client-side code libraries and tools, MOSAIC forms an SDK and application-development toolkit that can be applied to many vertical markets, including biomedicine, manufacturing, industrial design, cyberphysical systems, cybersecurity, and Software Language Engineering. This versatility is a result of the unique design requirememts for a platform that hosts data sets which are paired with self-contained, customized applications. Such requirements are, in essence: scientific data must be exposed in a representation that preserves semantically rich scientific principles; and dataset applications need a streamlined, modular design that limits external dependencies.

MOSAIC includes several new technologies that can be used and licenced separately or in combination, as listed below:

- ⋆ VersatileUX, a desktop client-side application framework;
- ⋆ NDP Cloud, a server-side cloud framework optimized for interoperating between desktop clients;
- ⋆ The MOSAIC Portal, itself, a new service for hosting and aggregating academic, scientific, and technical data and publications.

The MOSAIC portal bridges the gap between traditional academic publishing and the new world of software- and data-driven research platforms, which feature interactive reading experiences and open-access data sharing. While scientists have proposed new models for curating data and publications for a digital age — such as the Research Object Protocol, the FAIR (Findable, Accessible, Interoperable, Reusable) initiative, and FORCE (Future of Research Communication and e-Scholarship) — these paradigms have not been widely adopted by publishing companies. This in turn limits the market for new products that can truly enhance the reading and research experience. By bridging this gap, the MOSAIC portal can implement a deeper layer of academic, scientific, and technical data sharing than is currently offerred by publishers or online communities.

1 *The MOSAIC Architecture*

A MOSAIC portal combines a central web service with a collection of independent "Research Objects" (datasets bundled as a package of code and files whose structure conforms to the new Research Object Protocol), which are self-contained applications combining research data and application code. Not every document indexed by MOSAIC must have an associated Research Object, but the technological design of MOSAIC prioritizes integrating research portals with downloadable Research Objects.

In order to help authors create self-contained Research Objects, MOSAIC projects are structured around a specialized "*Reference Application Kernel*" (RAK), which presents a canonical format for bundling research data and code into a self-contained package. To be user-friendly, the MOSAIC Portal and tools make it easy to download the RAK bundle, and then to compile and launch the Dataset Application. In this fashion, readers can start exploring the data set with minimal hassle.

The RAK format includes other features as well, such as code export and testing, which provide a more detailed access to the published data (including several command-line executables built alongside the Dataset Application). However, the Dataset Application presents a useful introduction to the underlying data set, which helps users become oriented to the data set in its broad overview before they begin to examine the data set in finer detail.

Technically, MOSAIC RAK bundles are code repositories based on the Qt platform for C++ Applications. In RAK, each data set features C++ code which has no dependencies apart from Qt itself. Because Qt is an advanced desktop GUI development platform, it allows Research Objects to present compelling, interactive graphics and GUI

components — including 2D and 3D visualizations, tables and charts, context menus, dialog boxes, explanations of technical terms, embedded PDF viewers, and other features which make Dataset Applications (customized for each data set) an interactive and pedagogically useful tool for exploring raw data.

Research Objects indexed by MOSAIC do not necessarily have to use the RAK model. However, RAK provides a Reference Implementation demonstrating how Research Object metadata can be described for it to be automatically processed by a MOSAIC portal.

2 *The MOSAIC SDK and Data Modeling Features*

To facilitate implementation of RAK bundles, MOSAIC provides several development tools and code libraries which can help set up a development environment tailored to the MOSAIC ecosystem. These assets include the following:

Build Tools Since RAKs are designed to run inside Qt Creator, the RAK build system is based on *qmake*, which is Qt's layer atop the C/C++ "make" system. MOSAIC's build system extends *qmake* to simplify the integration of RAK applications with MOSAIC portals and with other scientific software that may be available as enhancements for RAK applications, depending on discipline/topics.

Hypergraph Code Libraries and Parsers MOSAIC defines a multiparadigm representation for scientific data based on Directed Hypergraphs. This representation is flexible enough to accommodate many different modeling paradigms — including conventional OWL Ontologies, Conceptual Spaces, and several other workflow and linguistics-based models — while staying conducive to self-contained Dataset Applications. The MOSAIC SDK provides code for creating hypergraphs from text documents and integrating hypergraphs into C++ applications.

Scripting and Testing The MOSAIC SDK includes components that developers can use to create test suites for Research Objects and to design a customized scripting environment. Selected functions from the RO code are exposed to a Runtime Reflection engine, which allows these functions to be invoked by scripts, including scripts composed for unit and integration testing. Functional annotations provide detailed Requirements Engineering for application procedures, including scientific details like statistical scale (Nominal/Ordinal/Interval/Ratio), dimensions, ranges, and declarations of side-effects.

MOSAIC Interop Applications built with the MOSAIC SDK can also automatically interoperate with MOSAIC portals. The SDK allows Dataset Applications to be equipped with special features for different user roles, including authors and editors as well as ordinary readers. Authorized users can automatically notify MOSAIC, from within their Dataset Application itself, about a new or updated version of their Research Object.

Architecturally, MOSAIC is developed as a Cloud/Native Hybrid featuring a collection of independent native applications (the RAK Research Objects) connected to a central cloud service (the MOSAIC portal). The portal itself is also a Cloud/Native Hybrid, in that a miniature version of a MOSAIC portal can run as a standalone desktop application. MOSAIC is implemented to allow seamless integration between native and cloud components — for example, using non-GUI Qt libraries as the core of the server-side code — so as to minimize data marshaling when the cloud service communicates with native endpoints.

3 *MOSAIC Server-Side and Client-Side Products*

Because of MOSAIC's unique design requirements — supporting a rich, multiparadigm semantics for scientific data, and centering a network of self-contained desktop applications — the MOSAIC SDK introduces innovative features that can improve application development methodology across a number of vertical markets: whereas the MOSAIC Portal is

a cloud service hosting an array of data set applications, a generic MOSAIC system can unify decentralized networks of self-contained desktop applications sharing data and services via a containerized cloud application.

The crucial feature of MOSAIC is that desktop-client endpoints are self-contained and architecturally lightweight — they are easy to download and compile — but are still Thick (Native/Desktop) Clients, equipped with a full arsenal of GUI, scripting, and testing features. For the MOSAIC Portal, these end-points are Research Objects presenting open-access research data in an interactive, reusable way. For other areas, MOSAIC facilitates the implementation of cloud-networked desktop applications that are flexible and versatile from a user's point of view, while also embodying transparent Requirements Engineering, rigorous code verification, and demonstrable compliance with legal and operational standards throughout the lifetime of a project.

NDP Cloud On the server side, NDP Cloud is a framework for deploying cloud services tightly integrated with native applications. NDP Cloud applications can be tested and developed as standalone desktop applications before being uploaded to cloud servers. NDP Cloud also eliminates most of the technological gap between web and desktop implementations: NDP Cloud adopts mostly the same code libraries and programming paradigms as desktop applications (such as Qt/C++), so that client-server interop becomes analogous to networking between two desktop applications. For this reason, NDP Cloud can be a good solution for developers who want to use cloud services to augment the capabilities of desktop software, enabling peer-to-peer messaging and seamless cloud backup and personalization, which enhance the desktop experience.

VersatileUX On the client side, VersatileUX is an GUI application framework which leverages the peer-to-peer and personalization capabilities of NDP Cloud. VersatileUX supports a highly modular approach to application development, with tight correlation between data structures and the GUI elements used to display them. Any VersatileUX software is a composite of semi-autonomous modules, each of which is responsible for reading or receiving data conforming to specific structures (from a file or a network) and presenting these data structures in specialized GUIs. GUI and data-structure components are bound together via strong typing. Because VersatileUX components are developed in isolation, testing and compliance verification is simplified for the application as a whole. At the same time, each VersatileUX module can design its own scripting and personalization features. In this way, the overall application can be granularly fine-tuned to the needs of each user.

Dataset Creator (dsC) Also available is a "Dataset Creator" tool for authors and publishers who would like to develop Research Objects using some of the techniques available through the MOSAIC SDK. Unlike the SDK, dsC is not specifically engineered to work primarily with a MOSAIC portal, so it can be adapted to different online platforms which host or index data sets. For developers, dsC provides templates for a MOSAIC RAK that runs as a self-contained application presenting a Research Object; developers can then insert additional code or files to interoperate with the applicable host platforms.

NDP Cloud and VersatileUX work together to promote personalization and interoperability. Individual desktop applications, built with the aid of VersatileUX, can connect to an NDP Cloud service so as to track user identity across application instances (or even across different applications), while working through the cloud service for data backup as well as to connect multiple users, for purposes of messaging and collaboration. VersatileUX modules can be fine-tuned via user preferences and application-specific data obtained from NDP Cloud services, updating application state through Runtime Reflection. Reflecting MOSAIC's emphasis on complex semantics for scientific data, NDP Cloud and VersatileUX interoperate according to sophisticated protocols for exchanging strongly-typed data structures between clients and servers and between clients themselves (via cloud messaging). The key difference between this Cloud/Native Hybrid architecture and conventional client/server architecture (including other Cloud/Native paradigms) is that both server-side and client-side code take full advantage of strong typing (selecting the best aspects of Functional and Object-Oriented programming). Moreover a *single* type system is sustained across all client and server components.