Flexible data model for analysis projects

Object-oriented and hierarchical, any level of data-analysis project complexity can be accommodated.

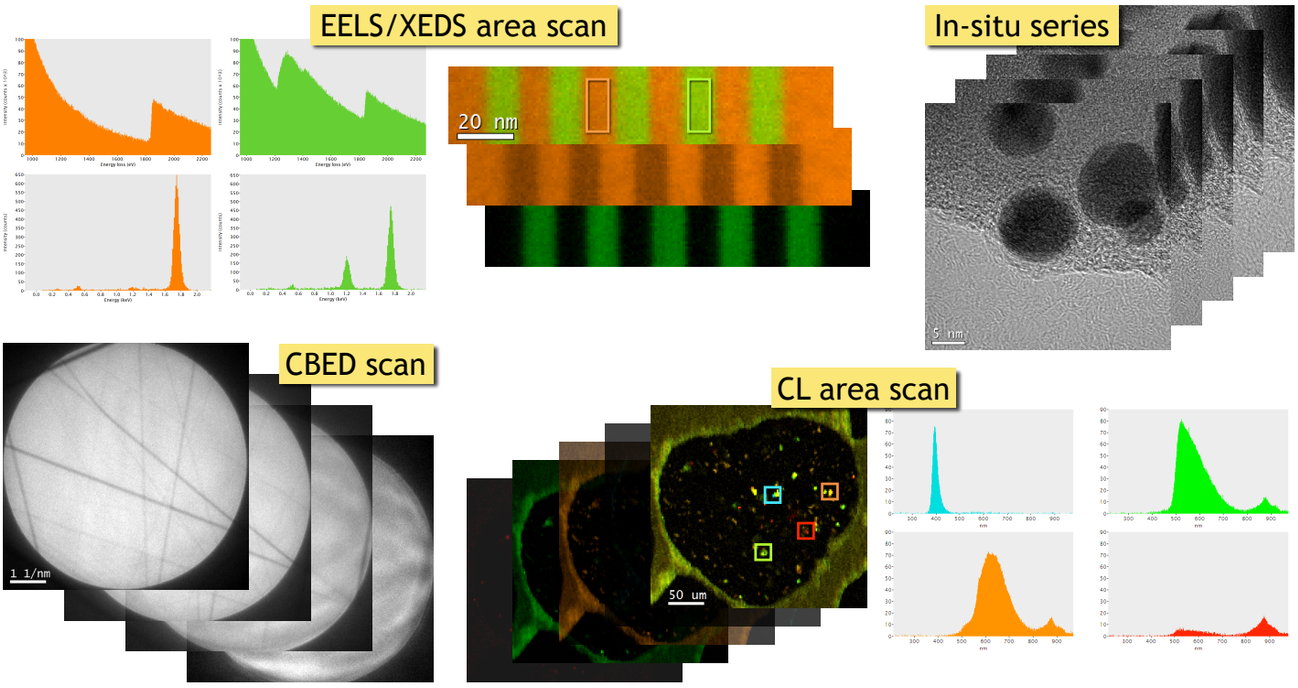

Figure 1 - Various examples of multi-dimensional data encountered in microscopy.

Background and motivation

Present-day microscopy and microanalysis techniques generate multi-dimensional and multi-component data sets, including hyperspectral XEDS, CL, and EELS mapping (aka spectrum imaging), scanned EBSD and CBED, confocal z-series, and in-situ time series (Fig. 1). The Enabler framework project seeks to establish a unified context to explore, integrate, and analyze such microscopy data sets captured with a broad range of techniques and instrumentation [1]. Enabler uses object-oriented (OO) design techniques [2, 3] to capture the concepts involved in thinking about, working with, and quantifying microscopy data in the form of ready-to-use software classes that can be deployed for a variety of data analysis applications.

The Enabler data model

A handful of classes form the basis of Enabler's general-purpose data model:

NumericBrick: N-dimensional numeric arrays and array math functions
PhysicalQuantity: physically dimensioned values (e.g feature size, spectral energy range, atomic density) with unit-propagating functions
PhysicalRange, PhysicalFrame: 1D and 2D selections on calibrated data
PhysicalBrickDatum: synthesis of the above into a complete, physically calibrated data object, with metadata and named data selections
SpatiallyLinkedDataSet: collection of datum objects with X and Y axes referenced to a common physical origin

Modular app architecture based on controllers

Controllers are re-usable software modules representing specific types of data objects and analysis tasks.

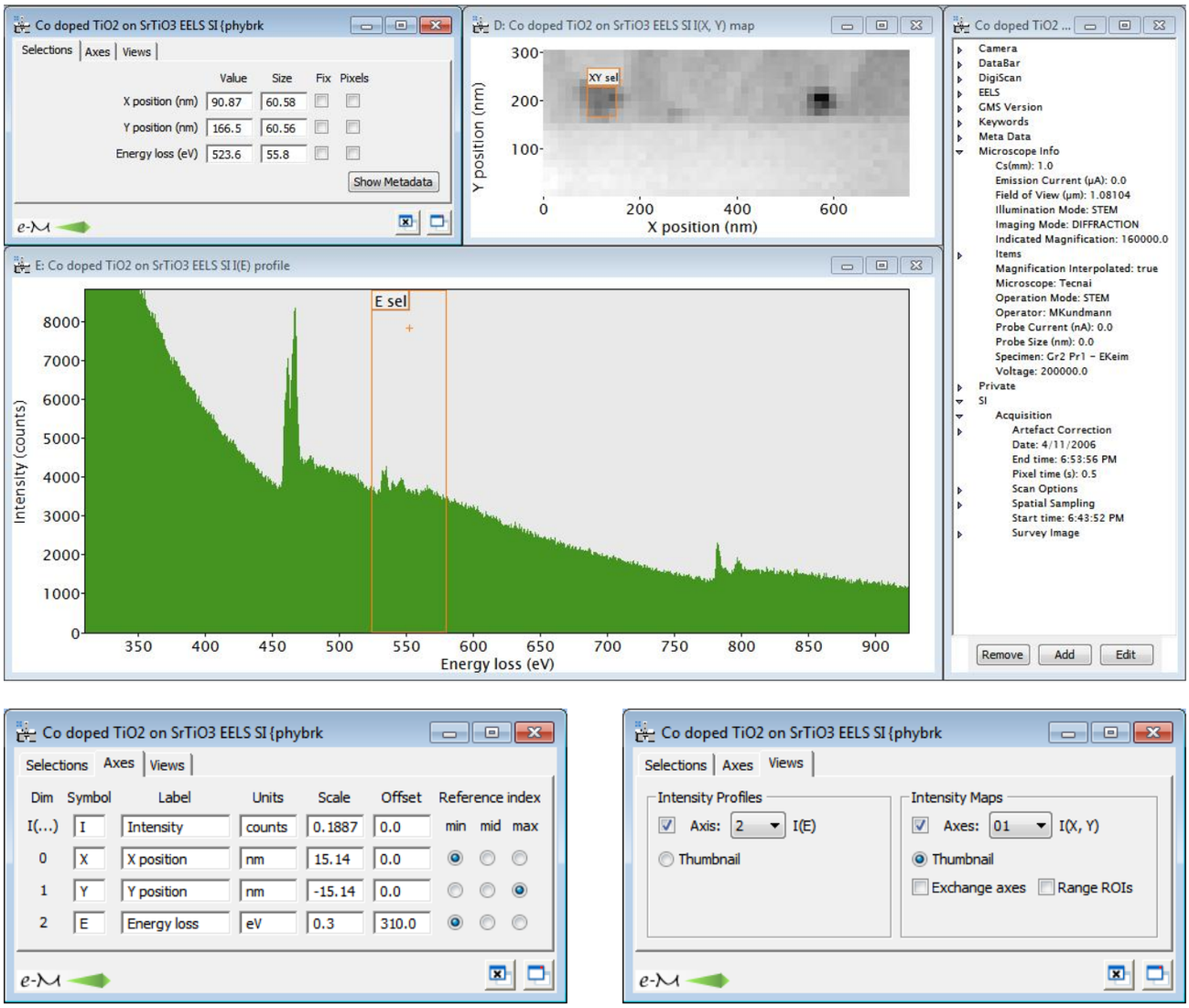

Figure 2 - An EELS spectrum image as an instance of a PhysicalBrickDatum MVC.

PhysicalBrickDatum and the MVC design pattern

The Enabler framework makes extensive use of the Model-View Controller (MVC) design pattern [4]. Controllers act as command dispatchers and mediators that keep subsidiary model data and views (UIs) in sync. In the Enabler framework, each controller is a complete program module, managing its data and parameters (including loading from and archiving to persistent storage), one or more task-relevant UIs (views), and associated algorithms. A key Enabler type for working with multi-dimensional microscopy data is PhysicalBrickDatumController (Fig. 2). Controller objects of this type fully encapsulate and provide access to the N-dimensional data array, its axis calibrations, and user-specified selections and views onto data slices and intensity profiles. An Enabler app designed to analyze a given type of data set can easily instantiate its PhysicalBrickDatumController from a disk archive and gain full access to its quantitative information via the method interface of the class. User inputs and adjustments are automatically re-archived.

A simple illustration: basic XEDS map extraction

Analysis project controller associates an X-ray signal range dictionary with an XEDS spectrum image datum controller.

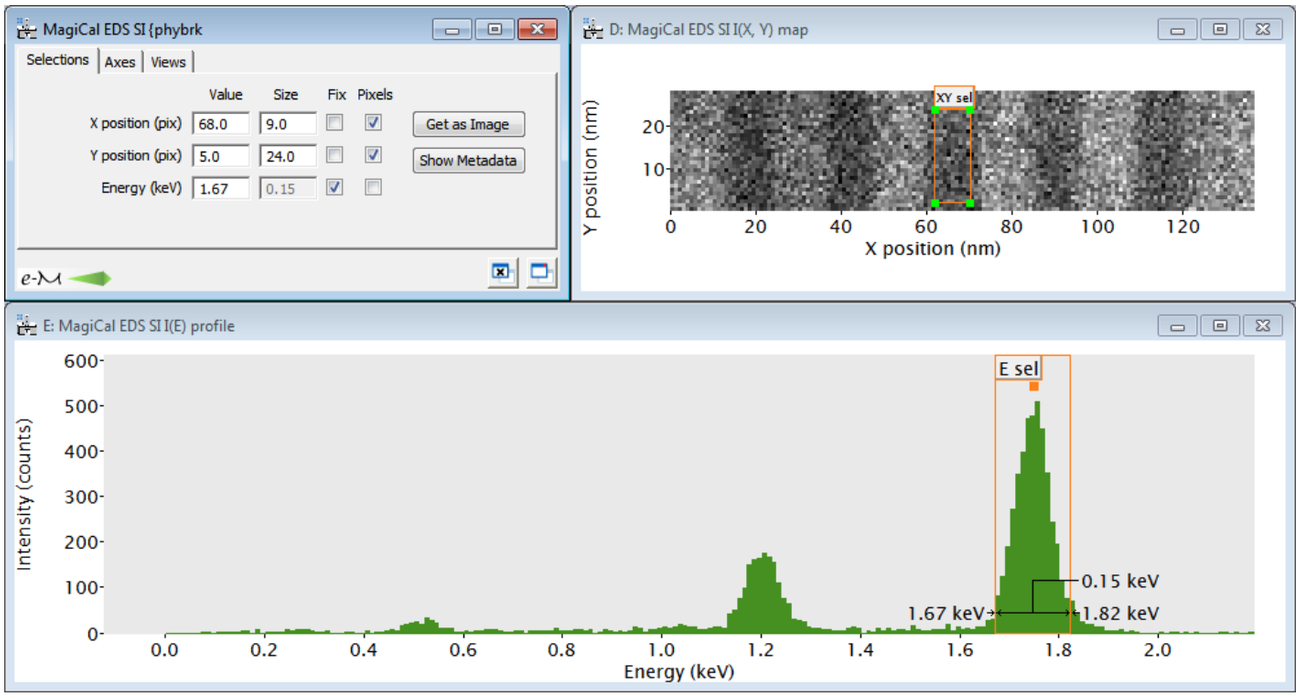

Figure 3 - An XEDS spectrum image (SI) as a PhysicalBrickDatum MVC.

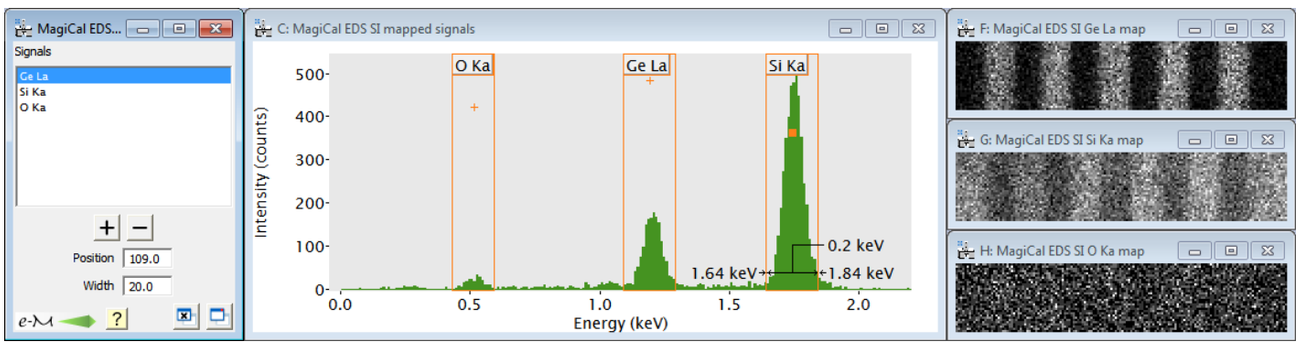

Figure 4 - A signal mapping project actively linked to the XEDS SI datum above.

Structure of an Enabler data analysis project

The basis of any data analysis project within the framework is yet another controller class derived from an Enabler MVC base class. For a basic XEDS map extraction module, the project controller needs a reference to an XEDS SI datum (Fig. 3) and a set (Dictionary) of named signal ranges that will be used to generate the signal maps. It requires a UI for user entry of the signals of interest and their ranges, as well as views onto the extracted maps (Fig. 4). The project and datum controllers are actively linked to each other so that user adjustments in one UI panel can be applied to other project components, as appropriate. The project controller, by implementing these linkages, provides all the task-specific intelligence needed to carry out the work of the project. A benefit of this approach is that code and data archive structures are closely linked, making Enabler project archives highly self-documenting and portable to other OO contexts.

Context for rapid data-analysis app development

Framework provides re-usable components that can be quickly deployed for microscopy-specific applications.

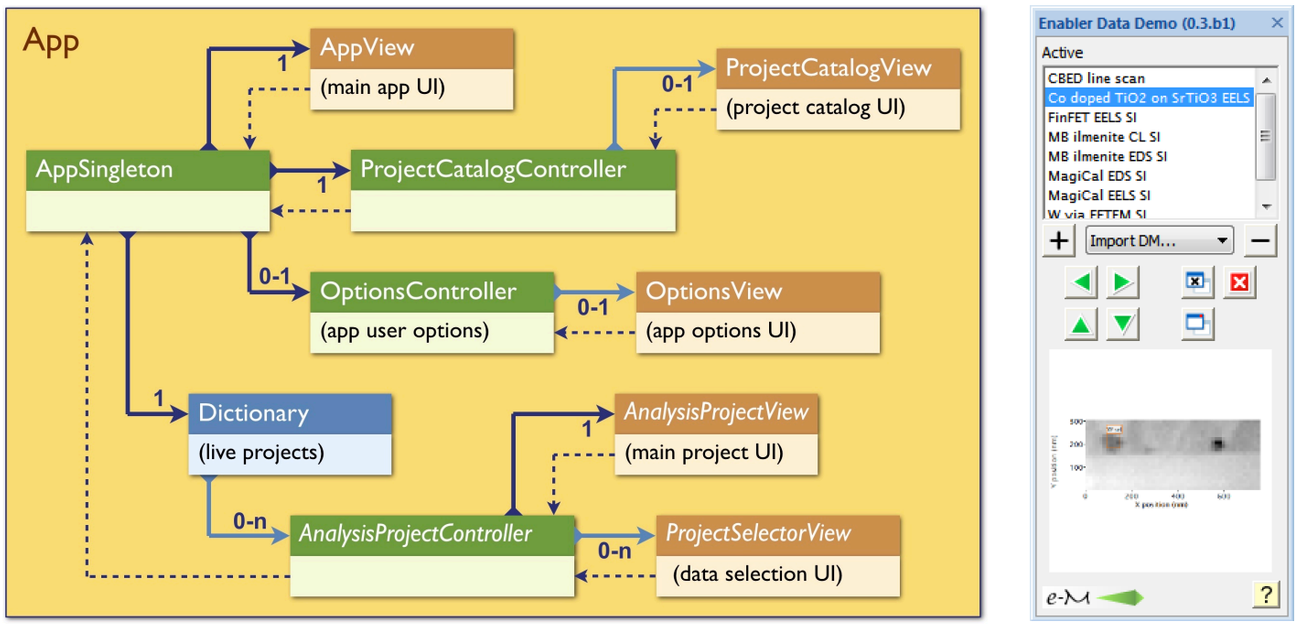

Figure 5 - Class diagram and view layout for an Enabler project-based app.

App architecture for data analysis projects

Within the Enabler framework, each analysis project consists of a tree or network of controller objects, where each controller must be instantiated and managed by a parent controller. In this way, events in a given controller can be passed up and down the controller hierarchy. Each such project tree must therefore be rooted in a root controller. This is the special role played by Enabler app modules (Fig. 5). Apps are singleton objects within the framework that are typically instantiated and managed by the host environment. This makes it possible to access any app object at runtime global scope, enabling apps to make use of each other's capabilities and subsidiary objects. The framework defines a design pattern for project-based apps that makes use of ready-to-use classes for the app UI, a project catalog MVC, and an app options controller. All of an app's option and preference data are archived to its own preference file in a user-specific AppData directory.

A reference document providing full details on the Enabler framework will be made available on the e-Metrikos website [5]. Parties interested in becoming Enabler users and/or developers should send email to info@e-metrikos.com.

References

[1] M. Kundmann, *Microscopy and Microanalysis*, **22** Suppl. 3, (2016) 290.
[2] G. Booch et al., *Object-Oriented Analysis and Design with Applications*, 3rd ed., Addison-Wesley, (2007) chs. 2 and 3.
[3] https://en.wikipedia.org/wiki/Object-oriented_programming
[4] E. Gamma et al., *Design Patterns*, (Addison-Wesley, 1994) p. 4.
[5] http://www.e-metrikos.com/enabler