Enabler: An Open Extendable Object-based Model for Microscopy Data Analysis

Michael K. Kundmann
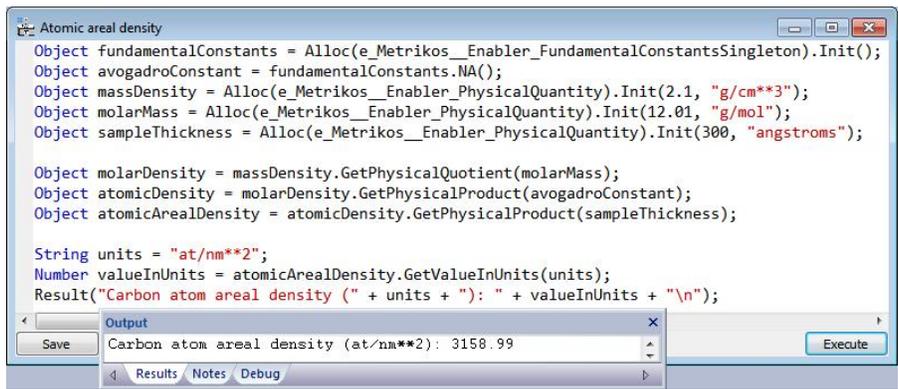
e-Metrikos, Pleasanton, USA

The Enabler framework project seeks to establish a unified context in which to explore, integrate, and analyze microscopy data sets captured with a broad range of techniques and instrumentation [1]. Enabler uses object-oriented design techniques [2, 3] to capture the concepts involved in thinking about, working with, and quantifying microscopy data in the form of fully functional, ready-to-use software classes that can be harnessed and deployed for a variety of data analysis applications. A handful of classes form the essential basis of Enabler's general-purpose data model: NumericBrick, PhysicalQuantity, PhysicalRange, PhysicalFrame, and PhysicalBrickDatum. NumericBrick encapsulates multi-dimensional array data (the core of any present-day microscopy data set) and its mathematical operations (e.g. arithmetic, numeric calculus, linear fitting). PhysicalQuantity models physically dimensioned values (e.g feature size, lattice spacing, spectral peak energy, atomic density) with unit-propagating functions (fig. 1). PhysicalRange and PhysicalFrame extend the PhysicalQuantity concept to 1D and 2D data selections. Most quantitative information extracted from a data set is one of the above physical types. PhysicalBrickDatum integrates all of the above concepts into a complete, physically calibrated data set, with metadata and named selection ranges and regions of interest. It is built on another key concept of the Enabler framework, the model-view-controller (MVC) design pattern [1]. PhysicalBrickDatum is a particularly important model-controller class within the framework. It not only encapsulates basic multi-dimensional data objects, but also handles all user interaction with the data, via its associated PhysicalBrickDatumView object, and manages all save/recall activity with its archive on disk (fig. 2).

Enabler defines a standard object archiving architecture that makes it possible to generate modular and hierarchical project archives of arbitrary complexity. Following lessons from the HMSA file format effort [4], it archives NumericBrick objects in separate (often large) binary files while member objects containing mostly descriptive and parameter data are archived in associated (much smaller) hierarchically organized files. The latter directly reflect the object hierarchy of the live project in memory. Enabler clearly distinguishes the concept of object type from that of file type, thus allowing objects to be captured in any of various commonly accessible and non-proprietary file formats, including HMSA, in particular. For example, a PhysicalBrickDatum archive has object type 'phybrk', which can appear on disk as a file named 'DatumName {phybrk.xml' or as 'DatumName {phybrk.gtg4', (i.e. a DigitalMicrograph™ TagGroup archive [5]).

The Enabler framework has so far only been implemented for the DM-scripting environment, but its concepts and archives are general enough to be deployed and accessed in other contexts as well. Enabler includes base classes for rapid project-based app development for both commercial (licensed) and community-serving (free) applications. The current release (version 3.2) is freely available to all interested parties upon request and detailed documentation about its app architecture and additional public class libraries will be made available on the e-Metrikos website [6].

References:

[1] M. Kundmann, Microscopy and Microanalysis, 22 Suppl. 3, (2016) p. 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] M. Kundmann *et al*., Microscopy and Microanalysis, 17 Suppl. 2, (2011) p. 860.
[5] http://www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software
[6] http://www.e-metrikos.com/enabler

Figure 1. DM script code illustrating use of PhysicalQuantity class to calculate the areal density of atoms in an amorphous carbon film.
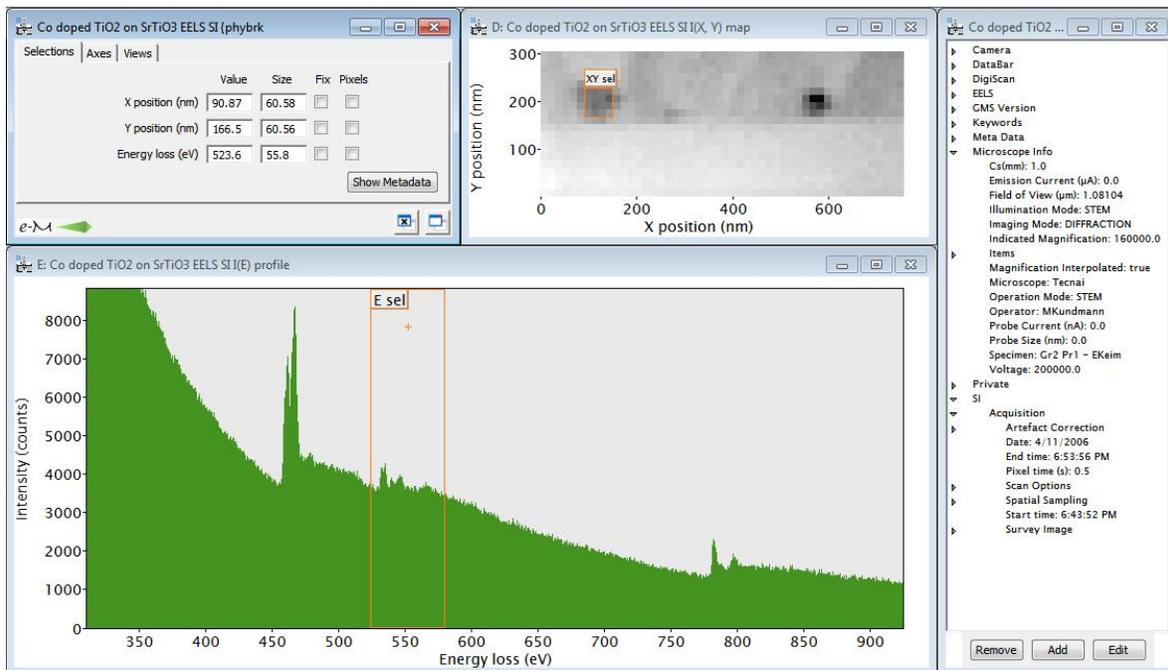

Figure 2. Live view onto PhysicalBrickDatum object (an EELS spectrum image). Adjustment of 'XY sel' or 'E sel' ROI triggers live update of spectral or image view, as well as the Selections numeric values. Changes are automatically saved to disk by methods of Enabler's self-archiving controller base class.