HDF File-based interoperability for multiscale modelling

Fabio Sacconi,
Stefano Bellocchio

Tiberlab Srl,
Rome, Italy

OUTLINE

- Deepen Project Overview
- Deepen Multiscale platform: Unicore implementation
- HDF choice as Deepen Common Data Format
  - ✓ HDF features
  - ✓ Implementation
  - ✓ Examples of use

From atom-to-Device Explicit simulation Environment for Photonics and Electronics Nanostructures

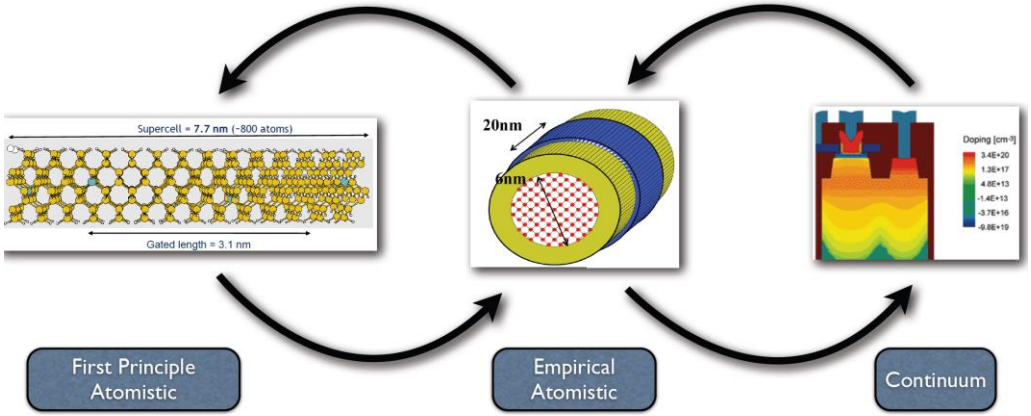
<u>Objective</u>:
Developing of an ***integrated open source multiscale simulation platform*** for predictive design of novel materials and nanostructures ***for CMOS and LED applications***

FP7-NMP Project DEEPEN, No. FP7-604416.

DEEPEN Challenges and Goals

New device simulators require

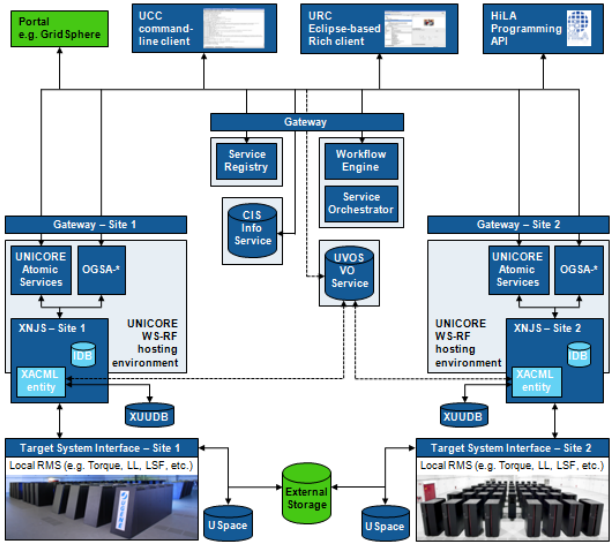
- An ***atomic-scale description*** of selected ***critical regions*** of advanced channel devices or nitride-based LEDs to capture details otherwise inaccessible.
- To resolve the considerable ***uncertainty in many critical parameters*** required for device optimisation.

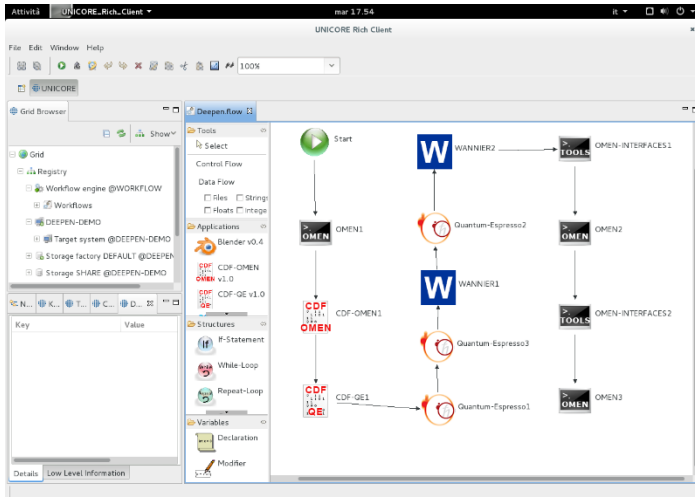


- DEEPEN addresses these challenges, coherently ***linking*** and ***coupling*** several state-of-the-art existing and newly developed tools for simulation of ***electronic properties*** and ***quantum transport***, integrated within a multiscale ***framework*** spanning from ***first-principles to macroscopic models***

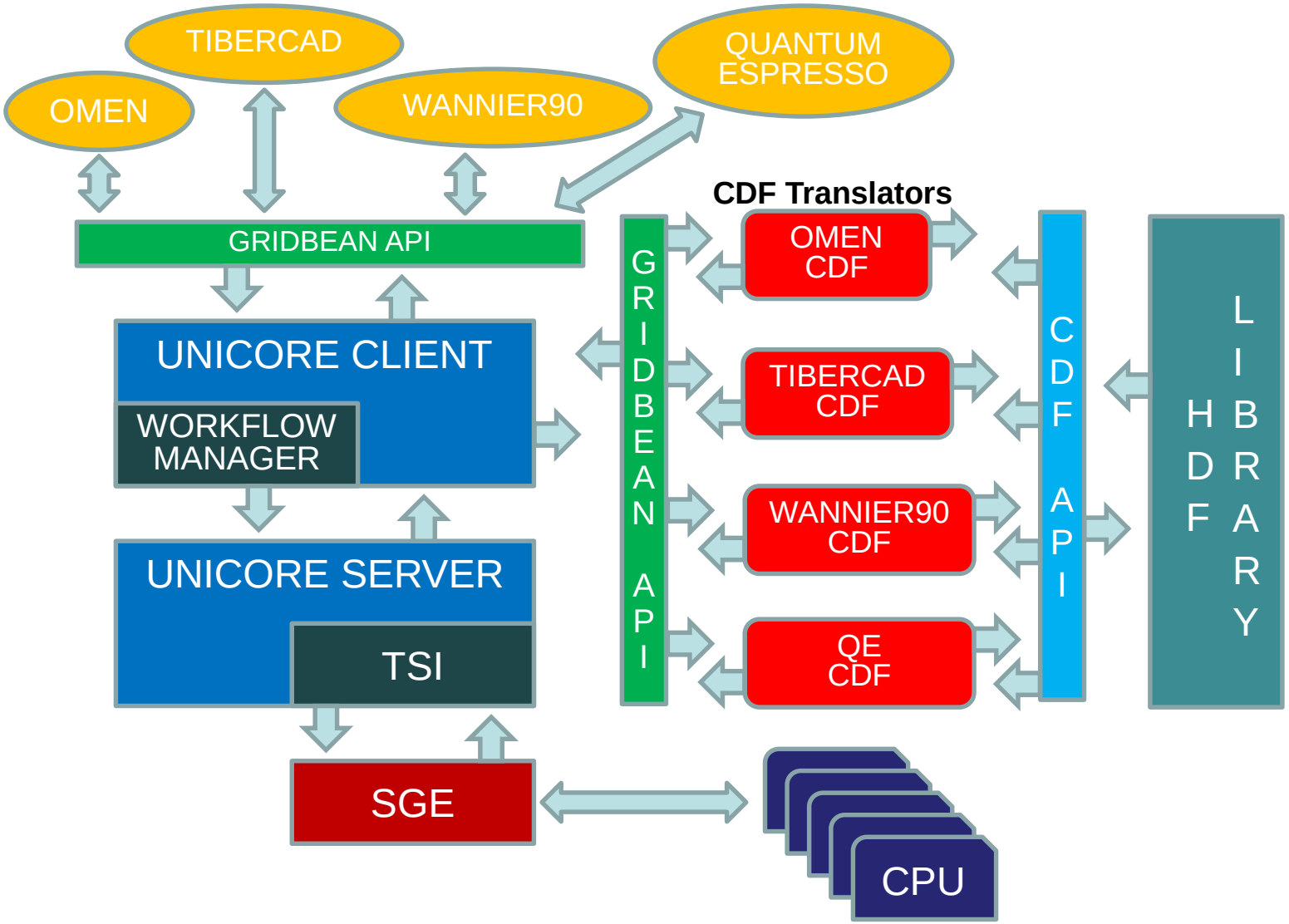
Deepen Platform

- Based on the **UNICORE** standard http://www.unicore.eu) for access to modern heterogeneous computer networks
- **Control of Simulation Flow** to be implemented using UNICORE clients for workflow generation
- **HDF**-based Open source Common Data Format (**CDF**) implemented to allow interoperability betweeen different simulation tools (DFT, Tight-binding, NEGF, Drift-Diffusion)
- ***CDF Translators*** and ***UNICORE APIs*** allow to exchange data while keeping application tools separated from the platform.
- **Metadata**: e-CUDS, e-CUBA in **HDF5** format

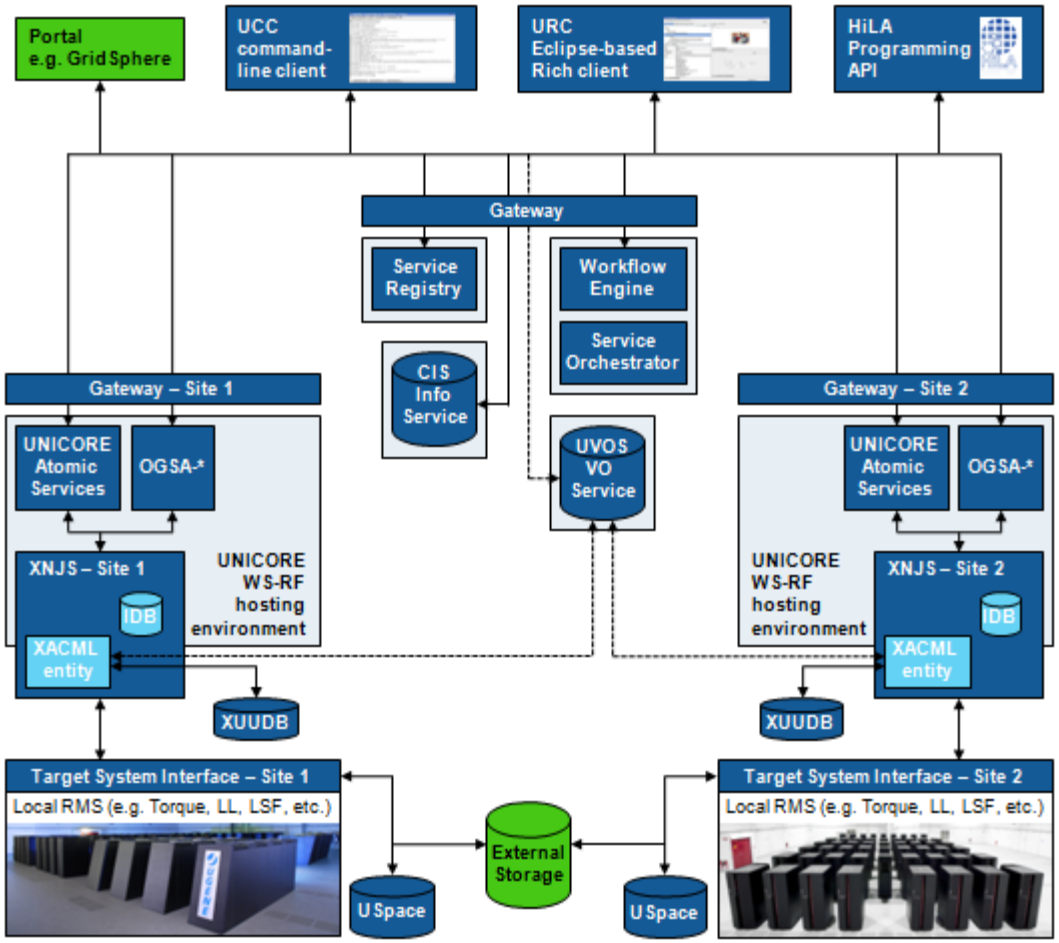




Deepen Platform

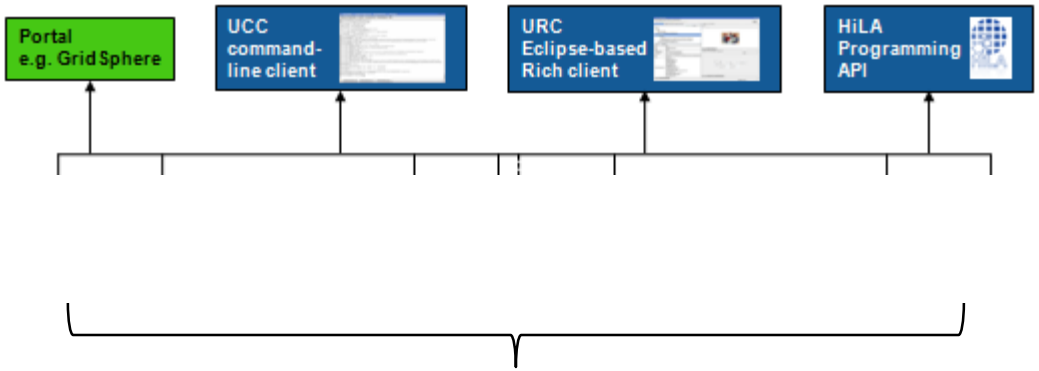


UNICORE – Schema Components



- **Client-Server Grid Middleware**
- Provides a seamless, secure, and intuitive access to different distributed computing and data resources
- *Open source* under BSD license.
- *Standards-based*, conforming to the latest standards from the Open Grid Forum (OGF), W3C, OASIS, and IETF
- *Open and extensible* realized with a modern Service-Oriented Architecture (SOA)

UNICORE – Schema Components



Client side

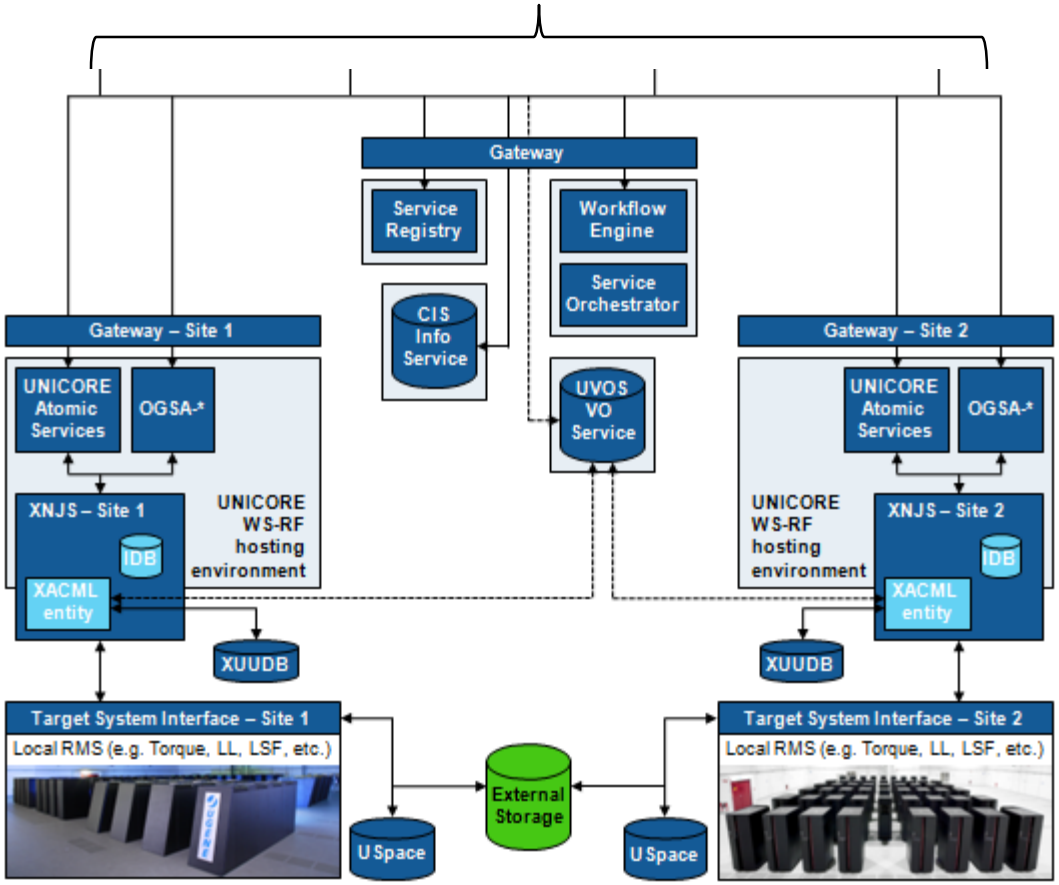
Generation of Workflows of the jobs to be submitted

Through different clients:

- Command-line (UCC)
- Graphic interface Eclipse-based (URC)
- APIs (HiLA)

UNICORE – Schema Components

Server side



The **UNICORE/X server** is the heart of a UNICORE site. Via web services it provides access to *storage resources, file transfer services, job submission* and *management services.*

- The **Gateway** component acts as the entry point to a UNICORE site and performs the authentication of all incoming requests. May serve several resources/target systems behind it.
- The **workflow engine** deals with high-level workflow execution
- The **service orchestrator** layer is responsible to executing the individual tasks in a workflow, handling job execution and monitoring on the Grid.
- In the **TSI** component the abstracted commands from the Grid are translated to system-specific commands. In a typical grid environment, a resource management software (***RMS***) is implemented.
- The **USpace** is UNICOREs job directory. A separate directory exists for every job, where the XNJS and TSI stores all input data and where stdout and stderr are written to.
- **External Storage** is a space accessible from Client and from TSI.

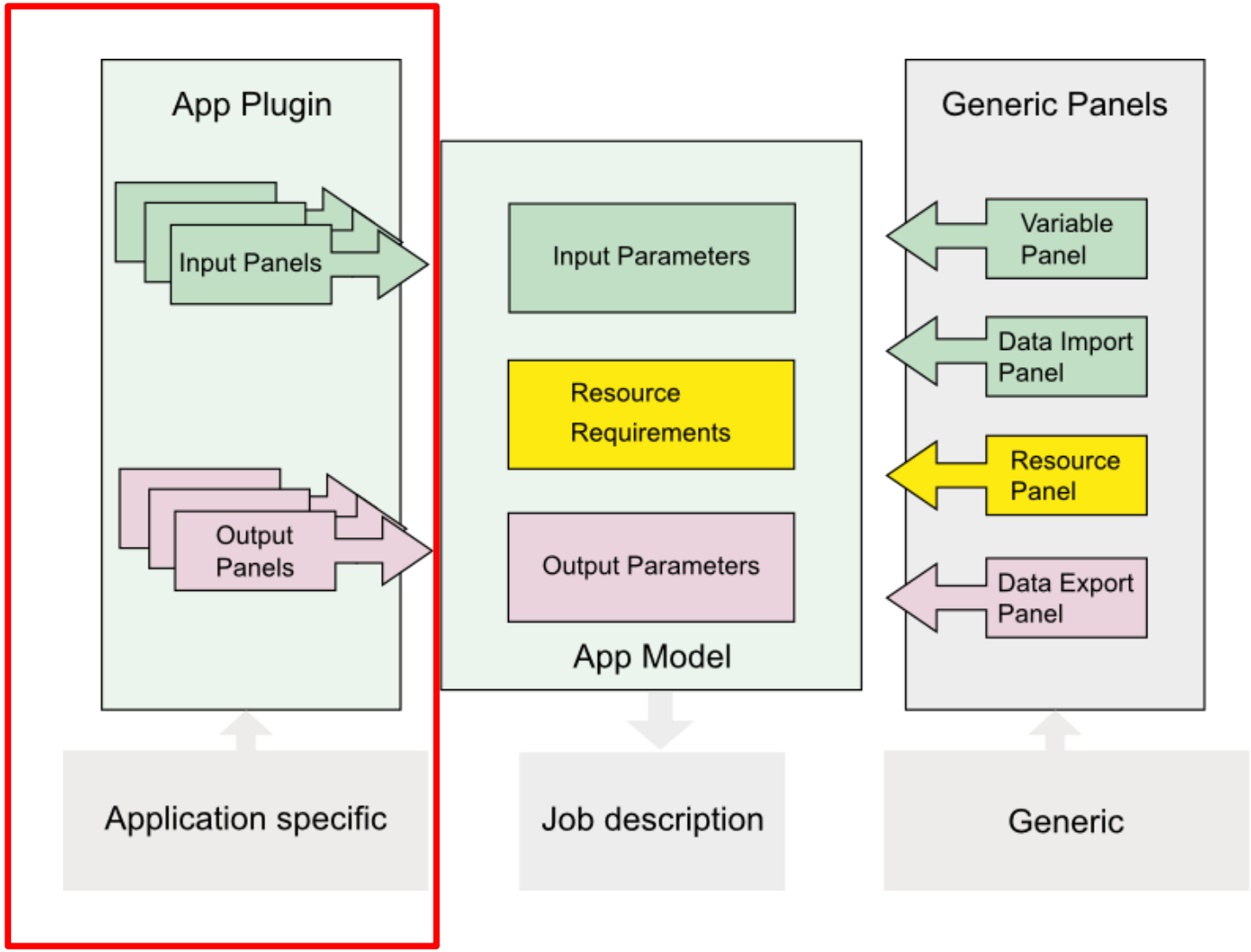
Unicore APIs (GridBeans)

- ***UNICORE APIs*** allow to exchange data while keeping application tools separated from the platform

- A UNICORE GridBean is an object responsible for:
  - ✓ generating job description for grid applications
  - ✓ providing graphical user interface for input data
  - ✓ providing graphical user interface for output data

- UNICORE GridBeans are divided into several modules:
  - ✓ a job description generation module
  - ✓ one or more user interface modules

GridBean - Overview



HDF in Deepen

A Common Data Format (**CDF**) based on the **Hierarchical Data Format** (HDF5) has been implemented in Deepen to allow interoperability betweeen different simulation tools

HDF5

Hierarchical Data Format (HDF)
Version 5

HDF software first developed in 1987 at the National Center for Supercomputing Applications (NCSA) at the University of Illinois

HDF Features

- A versatile **data model** that can represent very complex data objects and a wide variety of metadata.

- A software library that runs on a range of computational platforms, from laptops to massively parallel systems, and implements a high-level API with C, C++, Fortran 90, and Java interfaces.

- A **free and open source** (BSD license) general purpose platform for storing, managing, archiving, and exchanging data: thanks to **Open Format** HDF5 is widely supported in many softwares, including open source programming languages like R and Python, commercial programming tools like Matlab and open GIS tools like QGIS

HDF Features

- **Self-Describing**: Similar to XML, the datasets with an HDF5 file are self describing allowing an application to interpret the structure and contents of a file with no outside information

- Supports **Large, Complex Data**: HDF5 is a compressed format that is designed to support large, heterogeneous, and complex datasets. HDF5 format and software include features to store and access large datasets (even 1 TB in a file).
  - ✓ Efficient storage options, such as chunking, compression.
    Fast random access: optimized access to very large arrays, even when datasets are compressed
  - ✓ supports parallel I/O

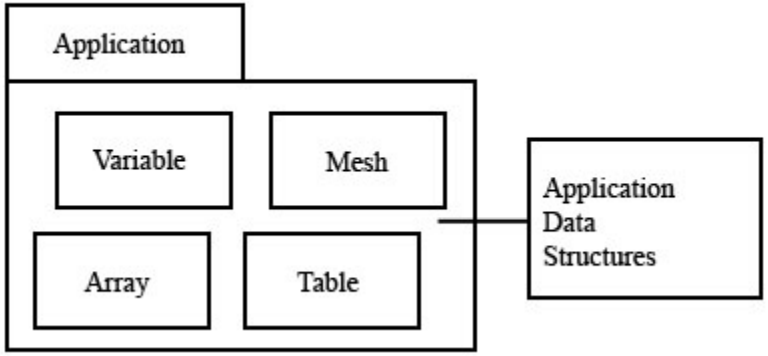
HDF Features

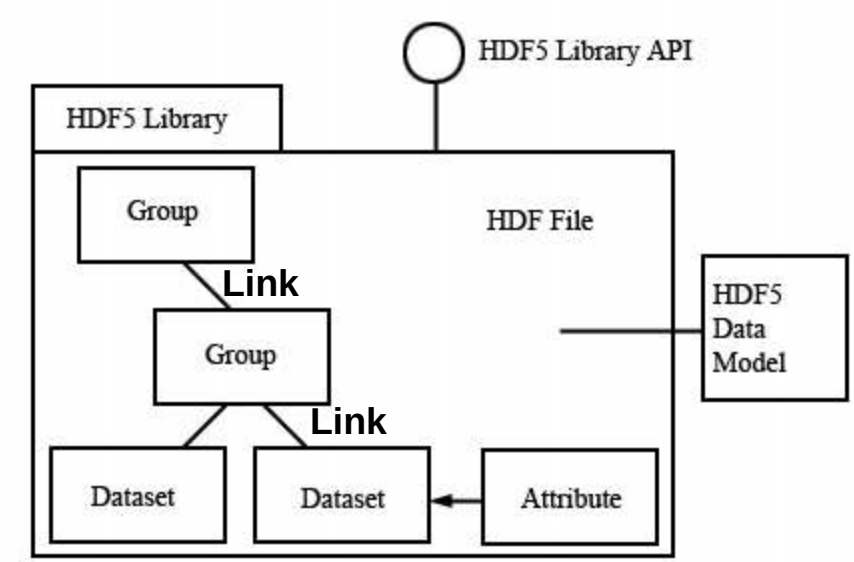
- Support **Heterogeneous Data**: Different types of datasets can be contained within one HDF5 file: tables, images, streams of data from instruments, and structured grids all in the same file. New objects can be easily added to existing HDF5 objects

- **Direct access** to parts of the file without first parsing the entire contents: "**Data slicing**", or extracting portions of the dataset as needed for analysis, means large files don't need to be completely read into the computers memory or RAM.

- **Hierarchical data ordering**: Supports complex data relationships and dependencies through the objects **Groups** (container structures) and **Datasets** (multidimensional arrays)

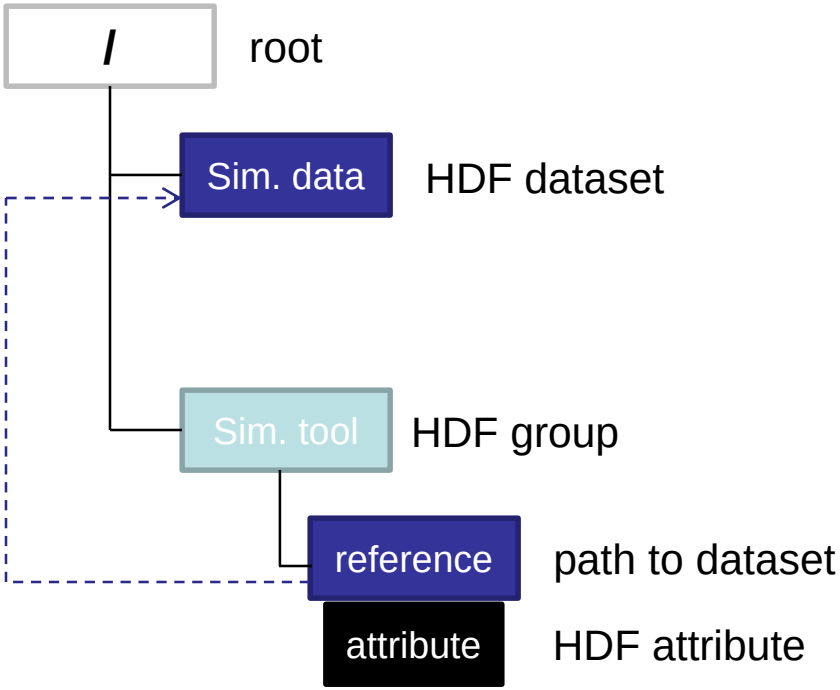
HDF: hierarchical structure

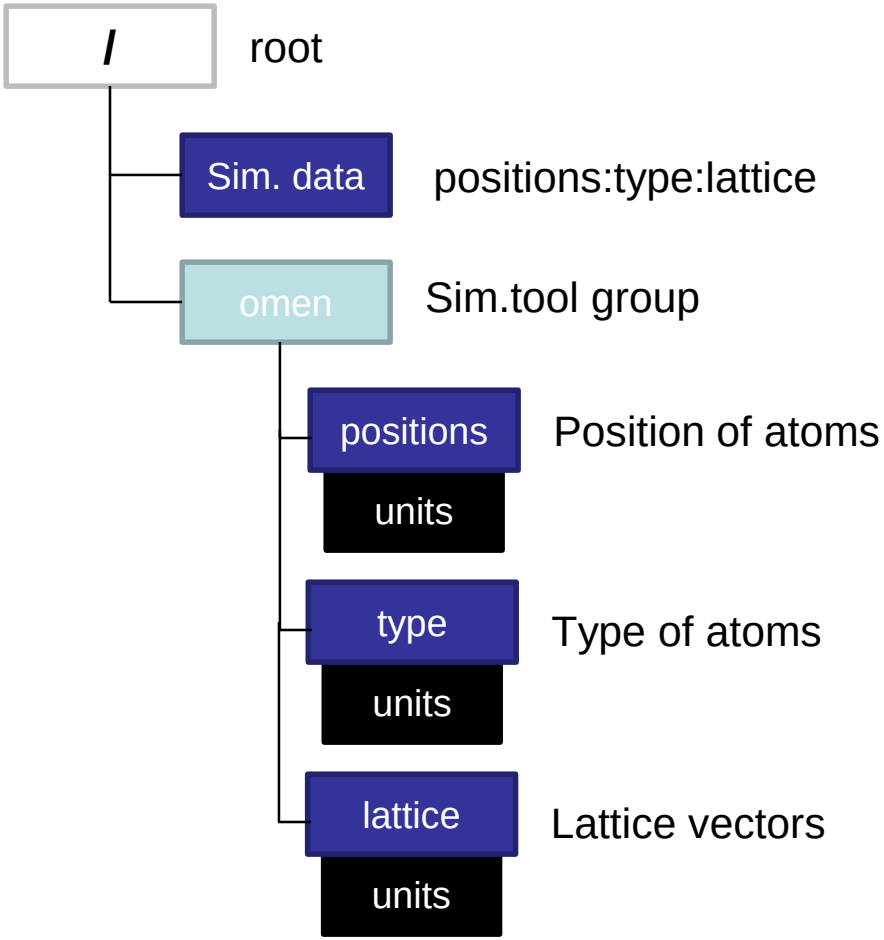




- *Group* - a collection of objects (including groups), analogous to a directory
- Group membership is implemented via *link* objects
- *Dataset* - a multidimensional array of data
- *Dataspace* - dimensions of a multidimensional array
- *Datatype* - a description of a specific class of data element

HDF implementation in Deepen

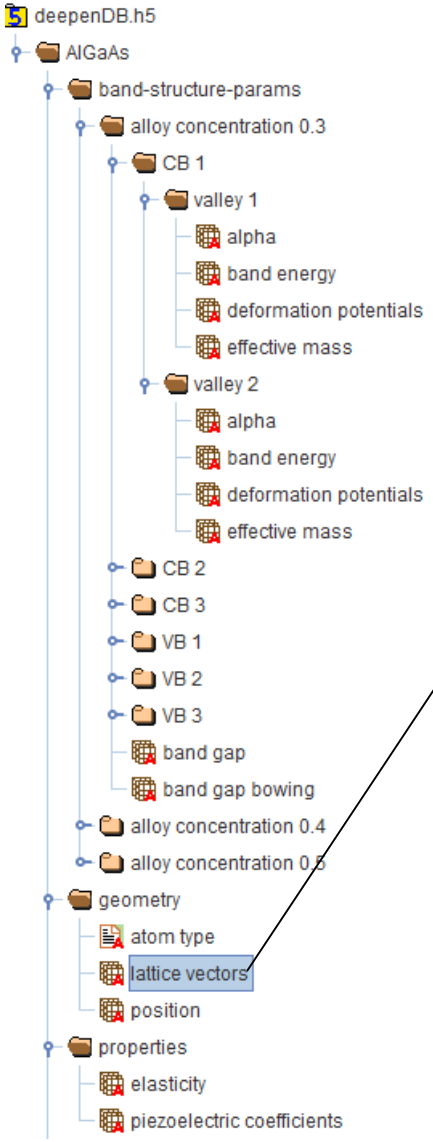




- HDF structured like a unix filesystem
- hierarchical representation of data, similar to a path

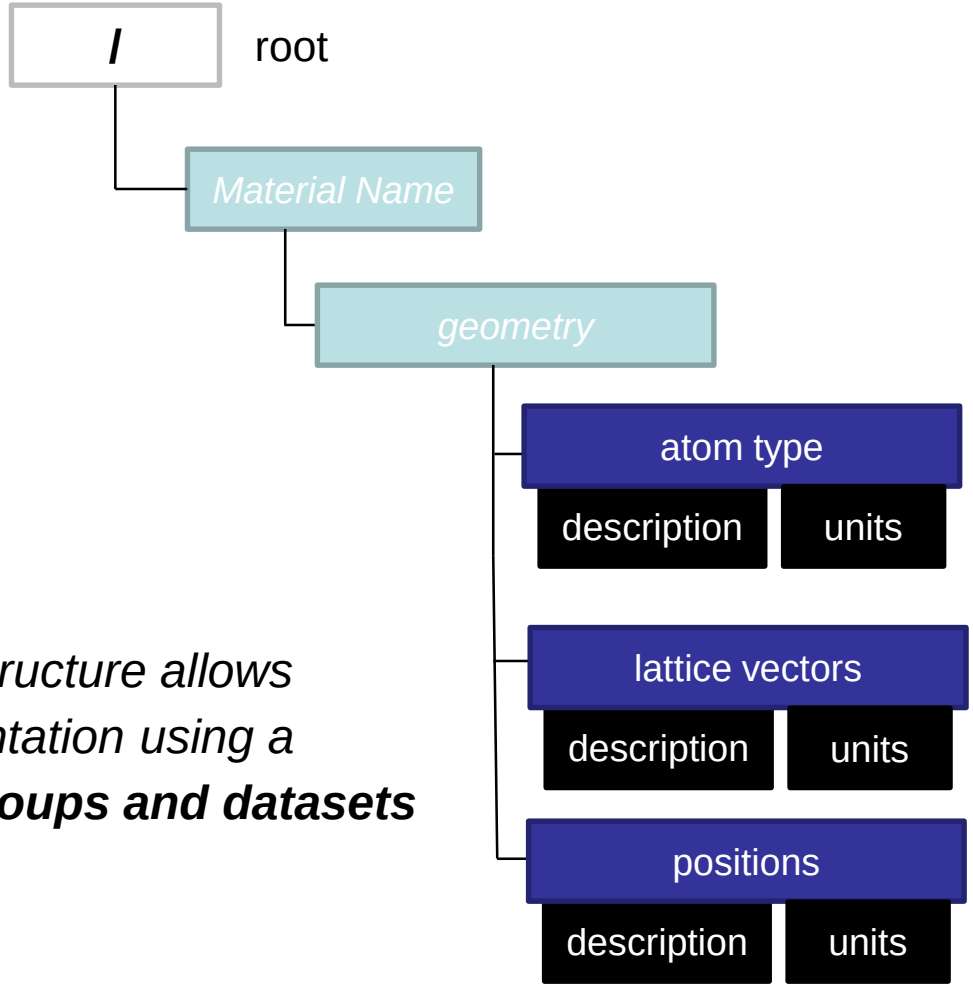
Example OMEN Diagram

HDF in Deepen – DataBase Structure



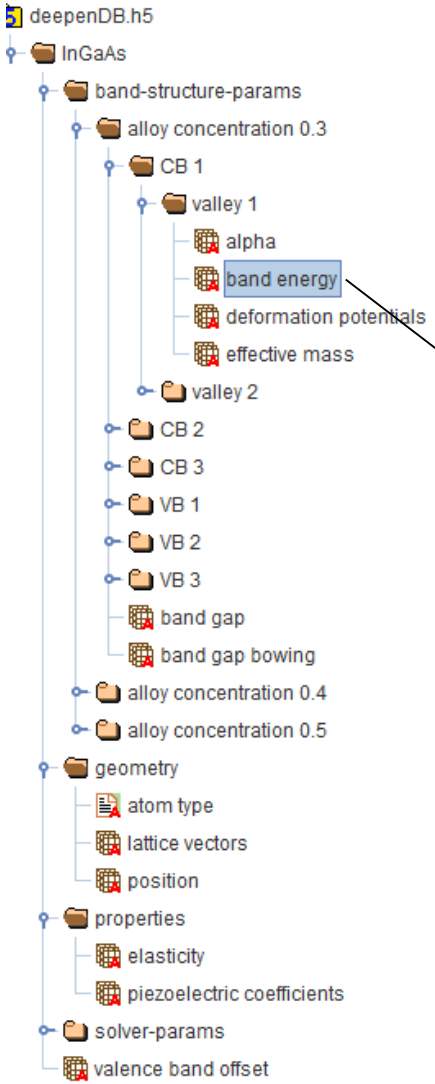
lattice vectors (22576, 4)
Array of 64-bit floating-point (3x5), 4
Number of attributes = 2
Description = lattice description
Units = double vector [nm]

HDF hierarchical structure allows database implementation using a ***combination of groups and datasets***

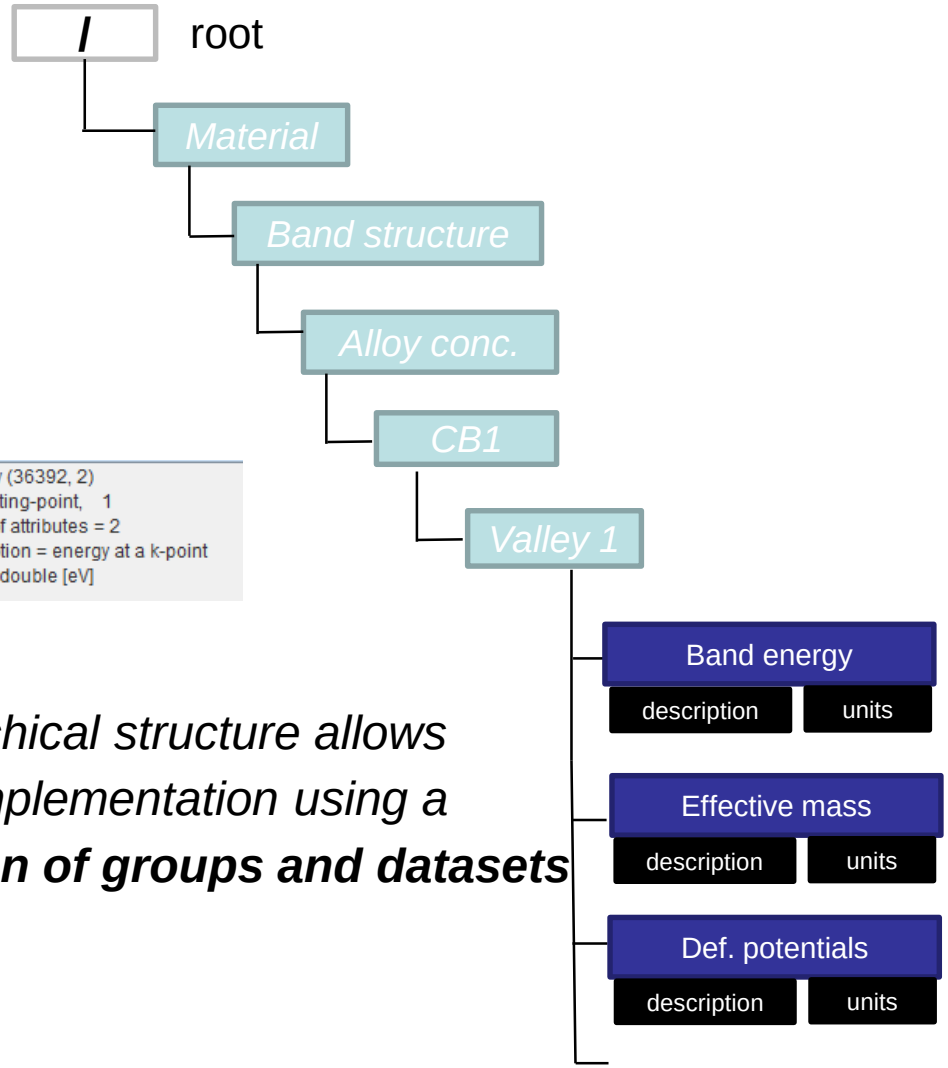


Example DB Diagram

HDF in Deepen – DataBase Structure



band energy (36392, 2)
64-bit floating-point, 1
Number of attributes = 2
Description = energy at a k-point
Units = double [eV]



HDF hierarchical structure allows database implementation using a ***combination of groups and datasets***

Example DB Diagram

Ab-initio transport: inter-changeable program flows

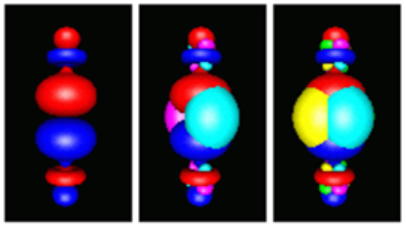
DFT Tool (VASP)

b-initio
ienna
ackage
imulation

Ab-initio electronic structure calculation

Interface available in Wannier90

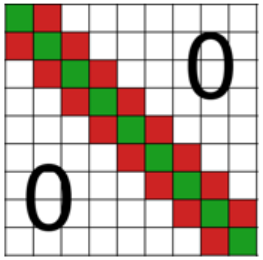
Wannier90



Conversion from plane-wave to MLWF basis

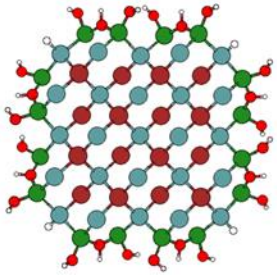
OSI

QT Tool (OMEN)



Quantum Transport

Calculation of current vs voltage in nanostructures



Welcome to OpenMX
Open source package for Material eXplorer

P O E N

Ab initio electronic Structure calculation

OSI

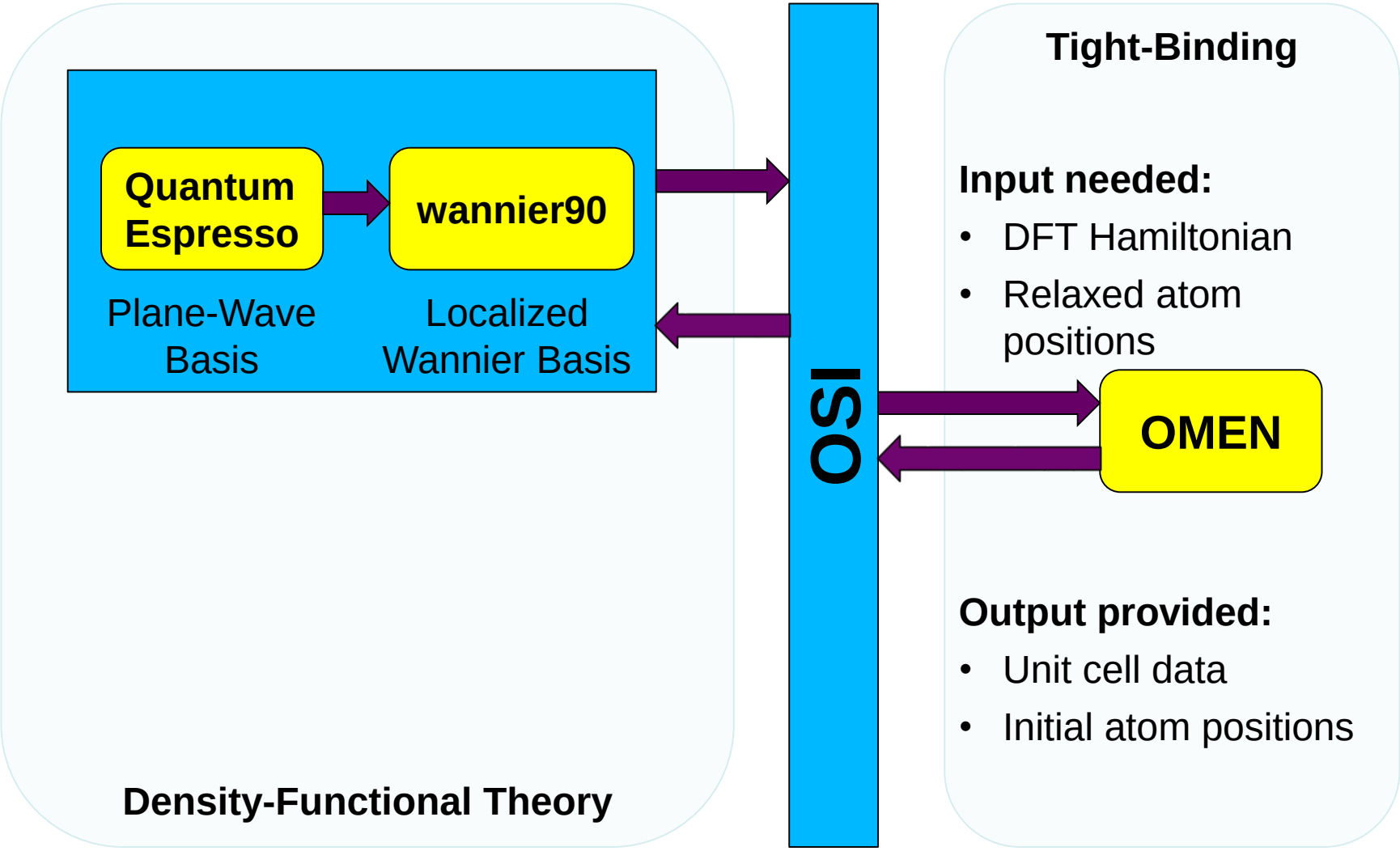
TiMeS

Quantum Transport

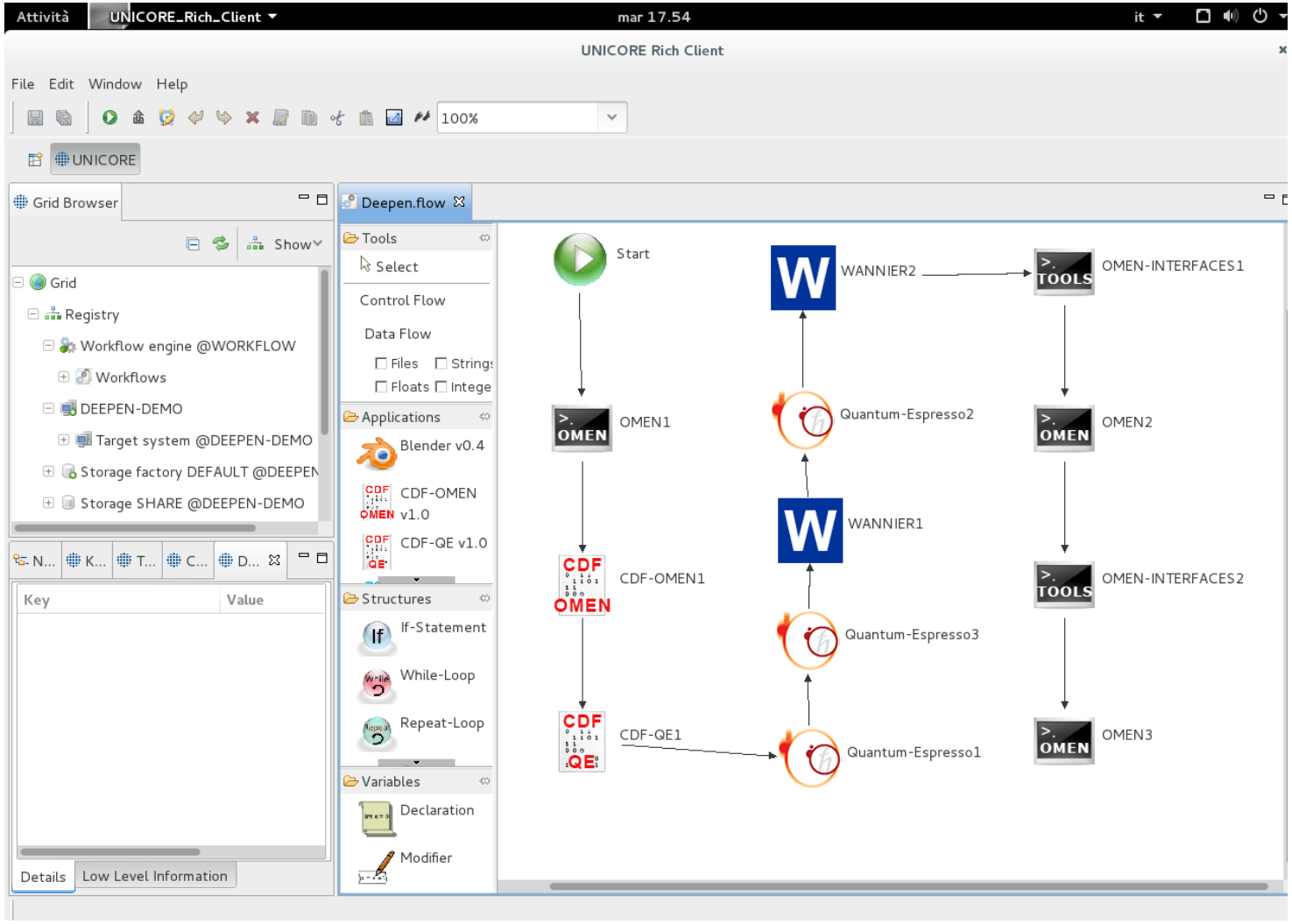
Tyndall National Institute

ETH Zürich

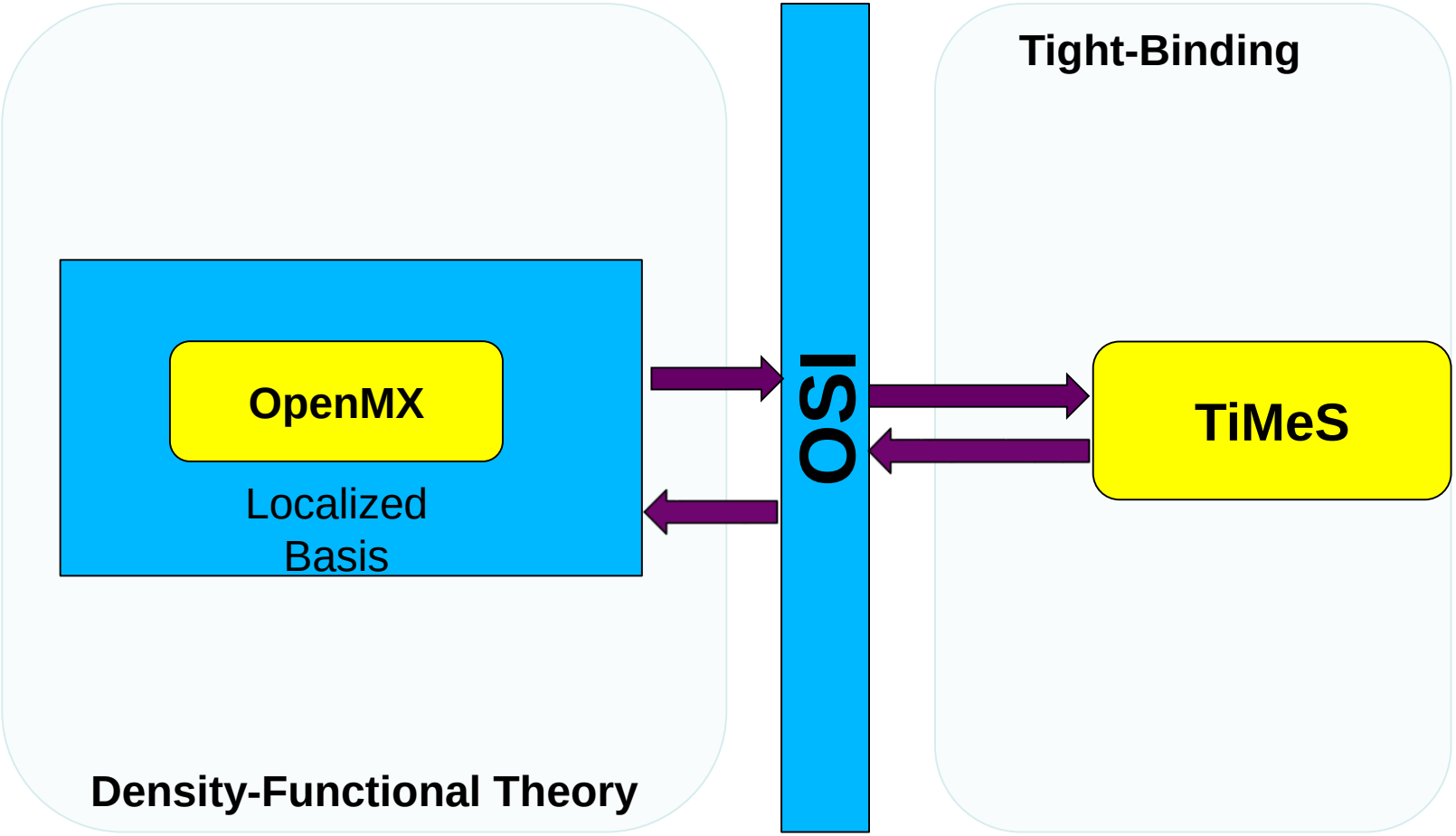
Test Workflow: DFT/Tight-Binding linking



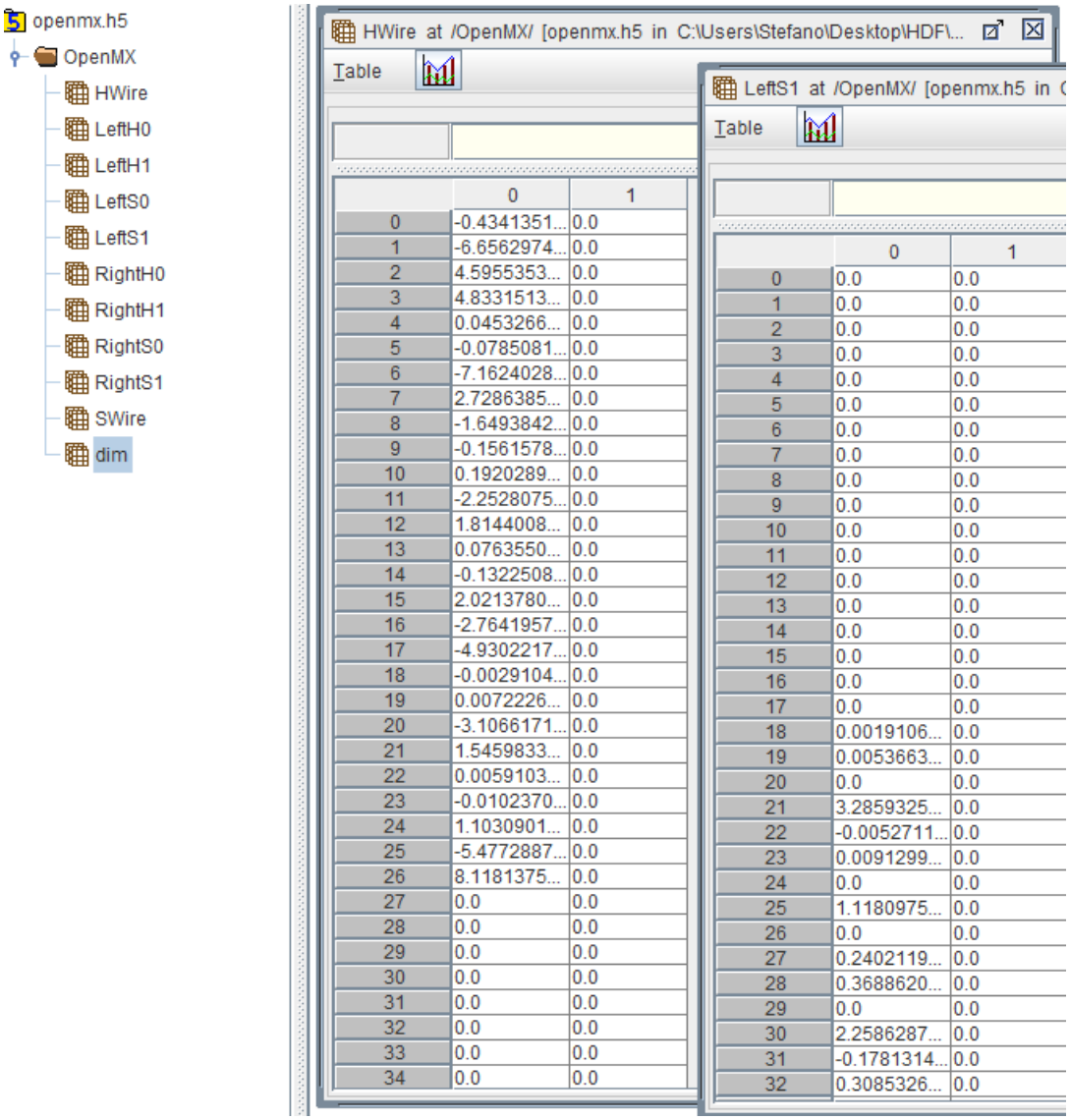
Test Workflow: DFT/Tight-Binding linking

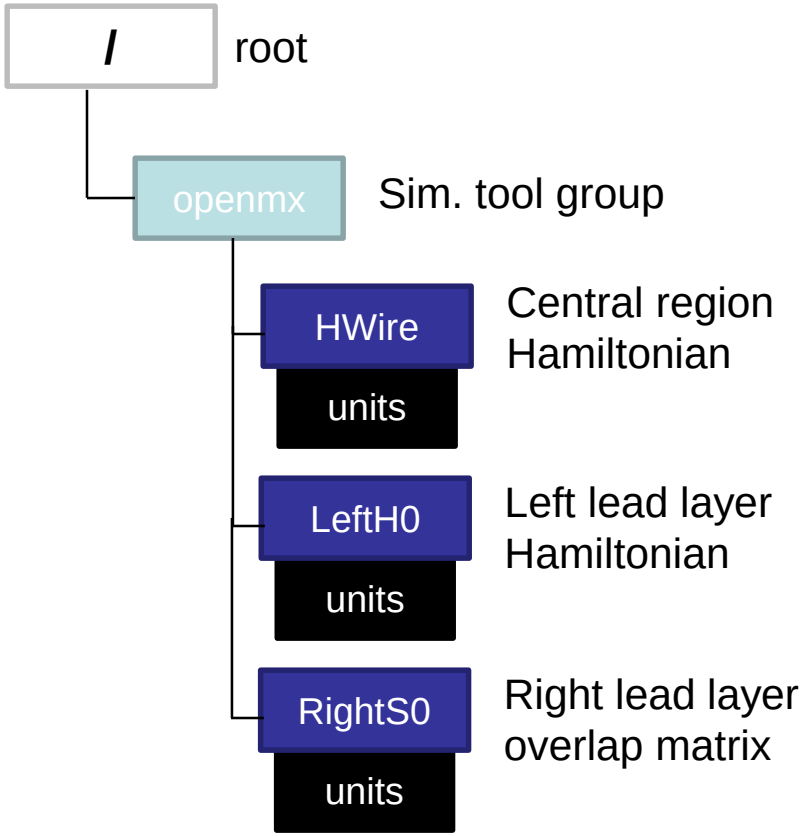


Workflow DFT/Tight-Binding linking in Deepen



HDF for model linking





Example OPENMX Diagram

Thank you!