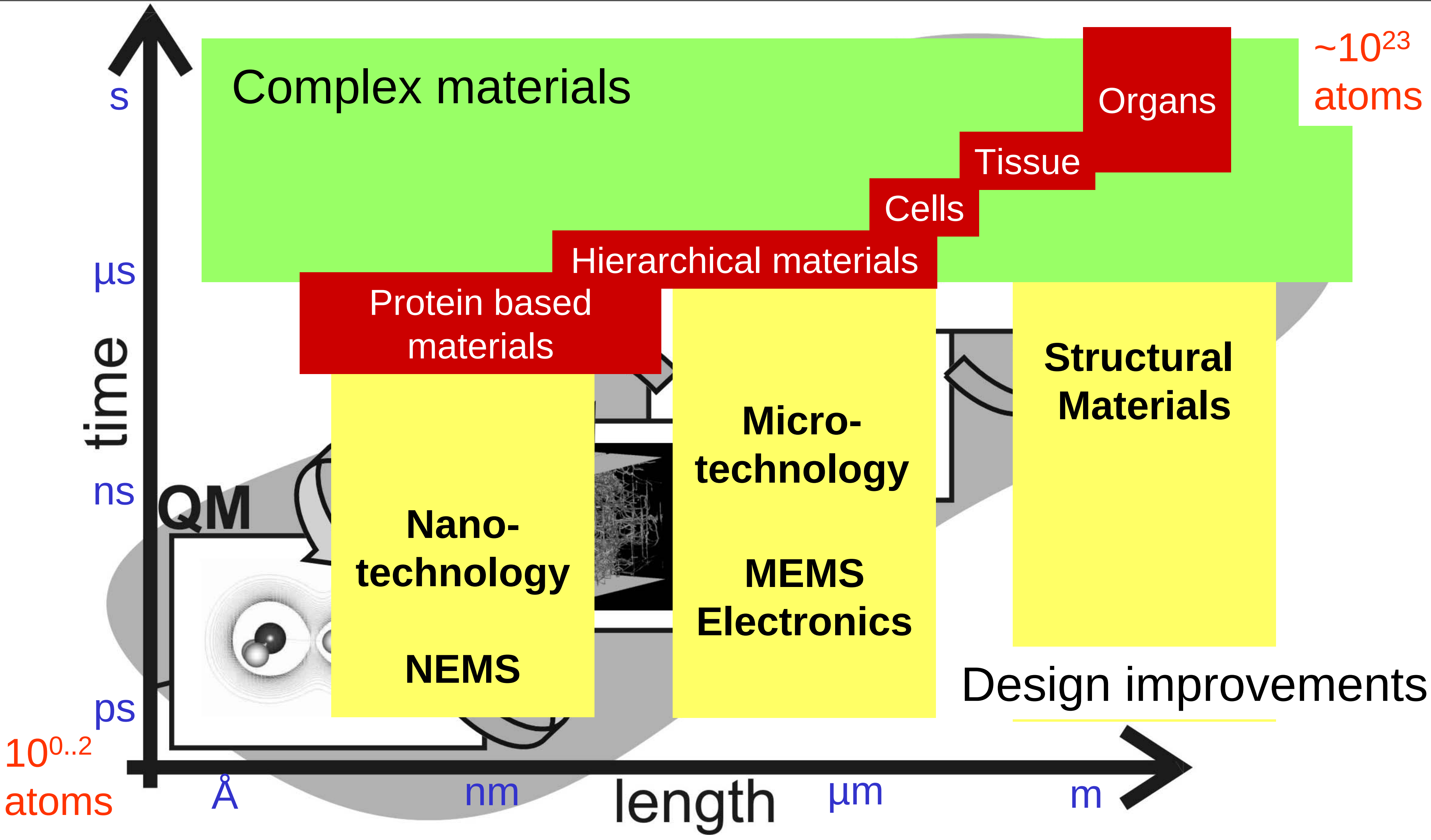
Challenges: Modeling engineering materials



Modeling complex materials is challenging: Need new modeling techniques

The vision for multi-scale modeling

- **Long-standing vision**
  Calculate macroscopic properties of materials by theoretical modeling or computer simulation from a very fundamental, *ab initio* perspective

- Possible strategy to solve this problem is to use methods based on distinct paradigms, operating at different scales

- This progress is possible with
  - The advent of efficient and accurate quantum mechanical methods (e.g. DFT),
  - Development of new empirical and semi-empirical potentials (EAM, ReaxFF…),
  - Enormous growth of computing power enabling studies with billions of particles.

- **Critical**: Breakthroughs in scale coupling techniques (e.g. QC method, reactive FFs) and analysis methods for complex systems (centrosymmetry technique)

Vision: Atomistic simulations of engineering properties at macroscopic scales to 1) understand fundamental mechanisms in materials (e.g. deformation, assembly), and to 2) predict properties of new materials to design new materials

Multi-paradigm multi-scale simulations

- For accurate prediction of materials properties based on first principles, need seamless integration of codes and methods ranging from quantum mechanics up to the continuum scale
- Most simulation methods: Only allow usage of a single method (e.g. EAM, DREIDING FF, QM…), rather than spanning a whole spectrum of state-of-the-art tools, including visualization & analysis
- **Paradigm complexity**: Whereas the last 20-30 years have been dominated by individual code and method developments, the time has come to integrate various methods into a concurrent environments
- A multi-method environment that allows scale-/ paradigm-agnostic combination of various simulation engines and simple additions to codes (e.g. new BCs) does not exist
- **We propose**: Development of multi-method multi-scale simulation tools allowing for seamless combination of various methods

Computational Materials Design Facility - CMDF

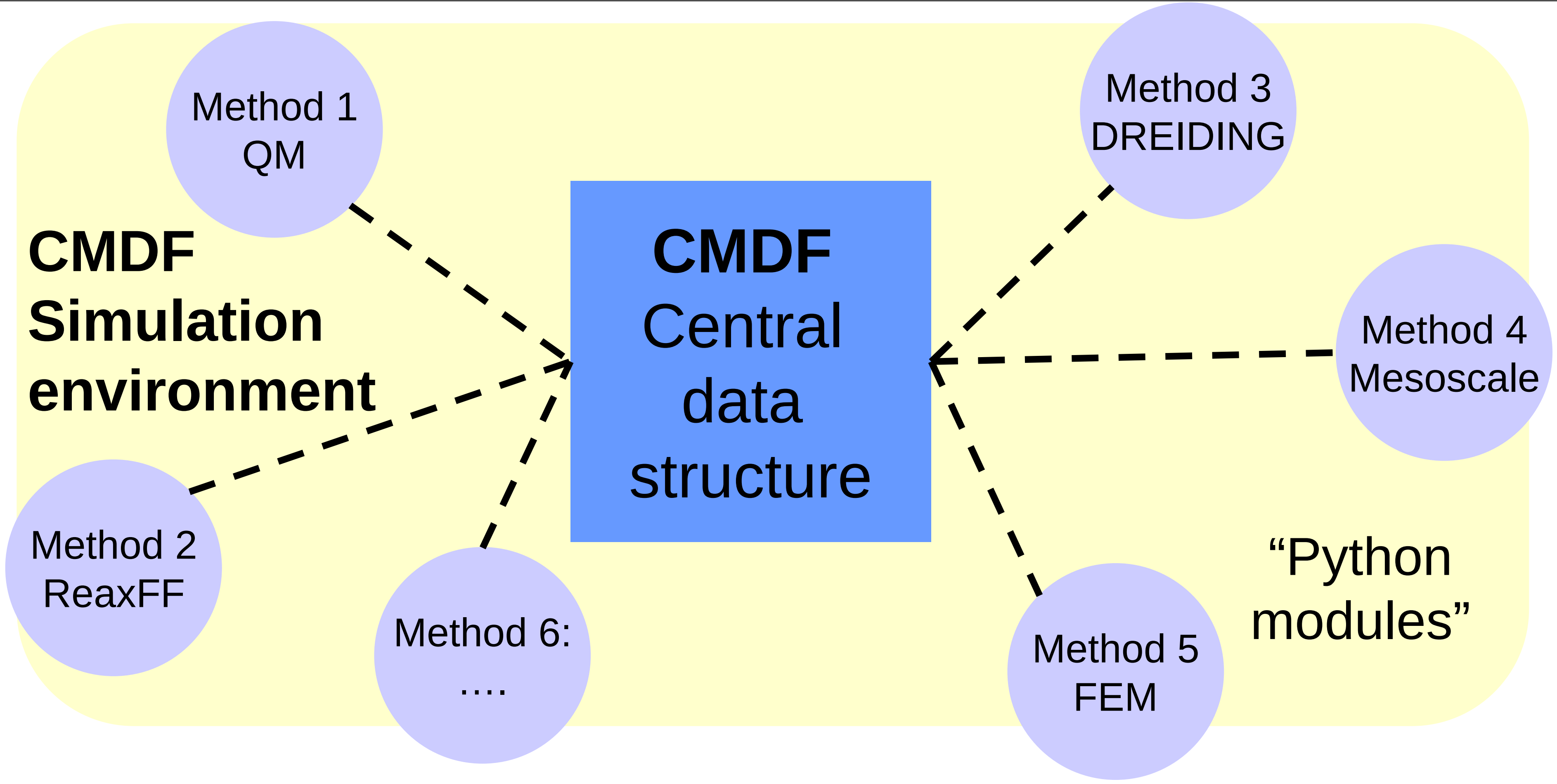
INTEGRATION MUST BE SCALE & PARADIGM AGNOSTIC, SIMPLE AND EXTENSIBLE

Historical perspective in scientific code development

- Until 1980s: FORTRAN and C codes
- 1990s: Beginning of scripting languages (with internet)
- 2000: Scientific computing with scripting languages
- Today: Multi-paradigm multi-scale simulations based on Python scripts

“*Paradigm complexity*”

How can this be achieved?

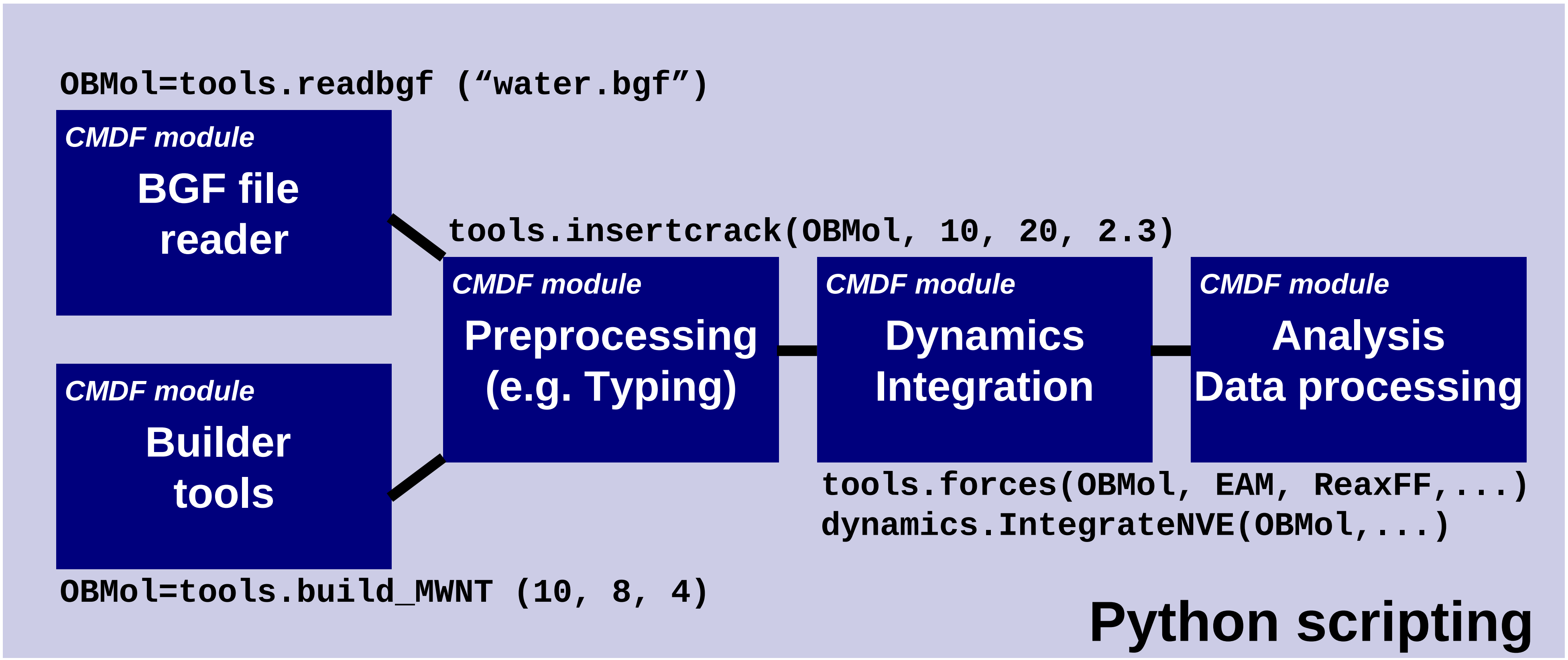


Central data structure can communicate with various modules

Provides: Simple and coherent access to complex simulation tools, enabling straightforward technology transfer and education of engineers in advanced computational methods

M.J Buehler *et al.*, MRS Proc. 2006

CMDF: A typical simulation script


OBMol=tools.readbgf ("water.bgf")
CMDF module
BGF file reader
CMDF module
Builder tools
OBMol=tools.build_MWNT (10, 8, 4)
tools.insertcrack(OBMol, 10, 20, 2.3)
CMDF module
Preprocessing (e.g. Typing)
CMDF module
Dynamics Integration
CMDF module
Analysis Data processing
tools.forces(OBMol, EAM, ReaxFF,...)
dynamics.IntegrateNVE(OBMol,...)
Python scripting

```
     OBMol=tools.readbgf ("water.bgf")
     OBMol=tools.build_MWNT (10, 8, 4)
     tools.insertcrack(OBMol, 10, 20, 2.3)
Loop {  tools.forces(OBMol, EAM, ReaxFF,...)
        dynamics.IntegrateNVE(OBMol,...)
     tools.printfracture (OBMol)
```

OBMol: ModBabel global data structure

Example CMDF Python script

```
import modbabel
import EAM_FF
import ReaxFF
import BUILDERMod
OBMol=tools.readbgf (“water.bgf”)
OBMol=tools.build_MWNT (10, 8, 4)
tools.insertcrack(OBMol, 10, 20, 2.3)

    tools.forces(OBMol, ???)
    dynamics.IntegrateNVE(OBMol)

tools.printfracture (OBMol)
```

```
tools.forces(OBMol, EAM)
```
or
```
tools.forces(OBMol, ReaxFF)
```
or
```
tools.forces(OBMol, Tersoff)
```
or
```
tools.forces(OBMol, DFT)
```

✓ Our scale agnostic design using the central data structure allows simple replacement of different simulation methods: Design methods that operate on OpenBabel objects

✓ This can be used to “try” different simulation engines during a simulation to find the most efficient method required to achieve a desired accuracy

M.J Buehler *et al.*, MRS Proc. 2006

CMDF fact sheet

- Ca. 35 different modules for various tasks including
  - File and data I/O (readers, file writers)
  - Energy and force calculation at different levels of detail
  - Pre- and post-processors (builders, analysis methods)
  - Filters and glues between different methods and scales
  - GUI for visual building
- CVS based code development and checkout
- Effort started around 2000 by Rick Muller
- Underlying codes in Python, C/C++ and FORTRAN codes seamlessly combined through wrappers, scripts and object code level integration
- Various modules can be used to build complex simulation tasks, such as multiple force field concurrent simulations, multiple scale/paradigm simulations,…

➢ Still under development but during the last year we have reached critical number of working (and “interfacable”) modules so that CMDF can be applied to real scientific problems

➢ **Also available through the Genepattern website**