The MolSSI Framework for Atomistic Simulations and Workflows

Paul Saxe, MolSSI

psaxe@vt.edu

https://molssi.org

Outline of talk

- Introduction to MolSSI
- Industry Needs
- Motivation for a workflow framework
- Workflow Framework
- Summary

Introduction to the Molecular Sciences Software Institute (MolSSI)

What is the MolSSI?

- Launched August 1st, 2016
- Funded by the National Science Foundation
- Collaborative effort by

Virginia Tech	T.D. Crawford
Rice U.	C. Clementi
Stony Brook U.	R. Harrison
U.C. Berkeley	T. Head-Gordon
Stanford U.	V. Pande
Rutgers U.	S. Jha
U. Southern California	A. Krylov
Iowa State U	T. Windus

12 Software Scientists
8 currently, 2 more hired and 2 open positions


Doaa Altarawy


Paul Saxe


Eliseo Marin-Rimoldi


Taylor Barnes


Ben Pritchard


Daniel Smith


Jessica Nash


Jonathan Moussa

21 Software Fellows

- 6 month initial phase
- Possibility of further 18 months
- Open to graduate students and postdocs at US institutions
- Next call mid-August

What is the MolSSI?

- Joint support from several NSF divisions:
  - Advanced Cyberinfrastructure (ACI)
  - Chemistry (CHE)
  - Division of Materials Research (DMR)
- Designed to serve and enhance the software development efforts of the broad field of computational molecular science.

Industry Needs

How do we find out?

Innovation. How to?

RAMP
THE ROANOKE-BLACKSBURG ACCELERATOR

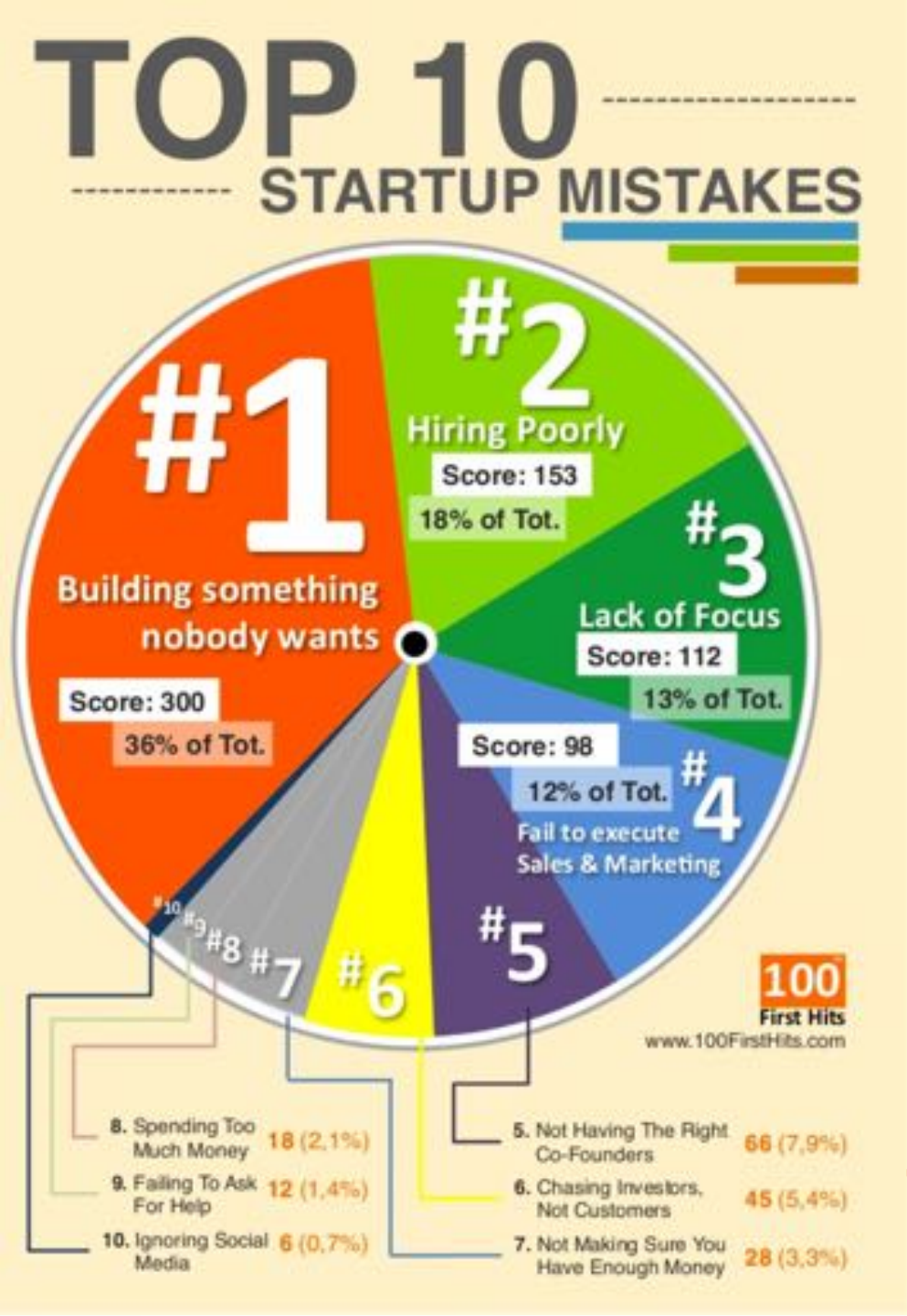
I-CORPS™
NSF Innovation Corps

Mike Abbott

Lisa Garcia

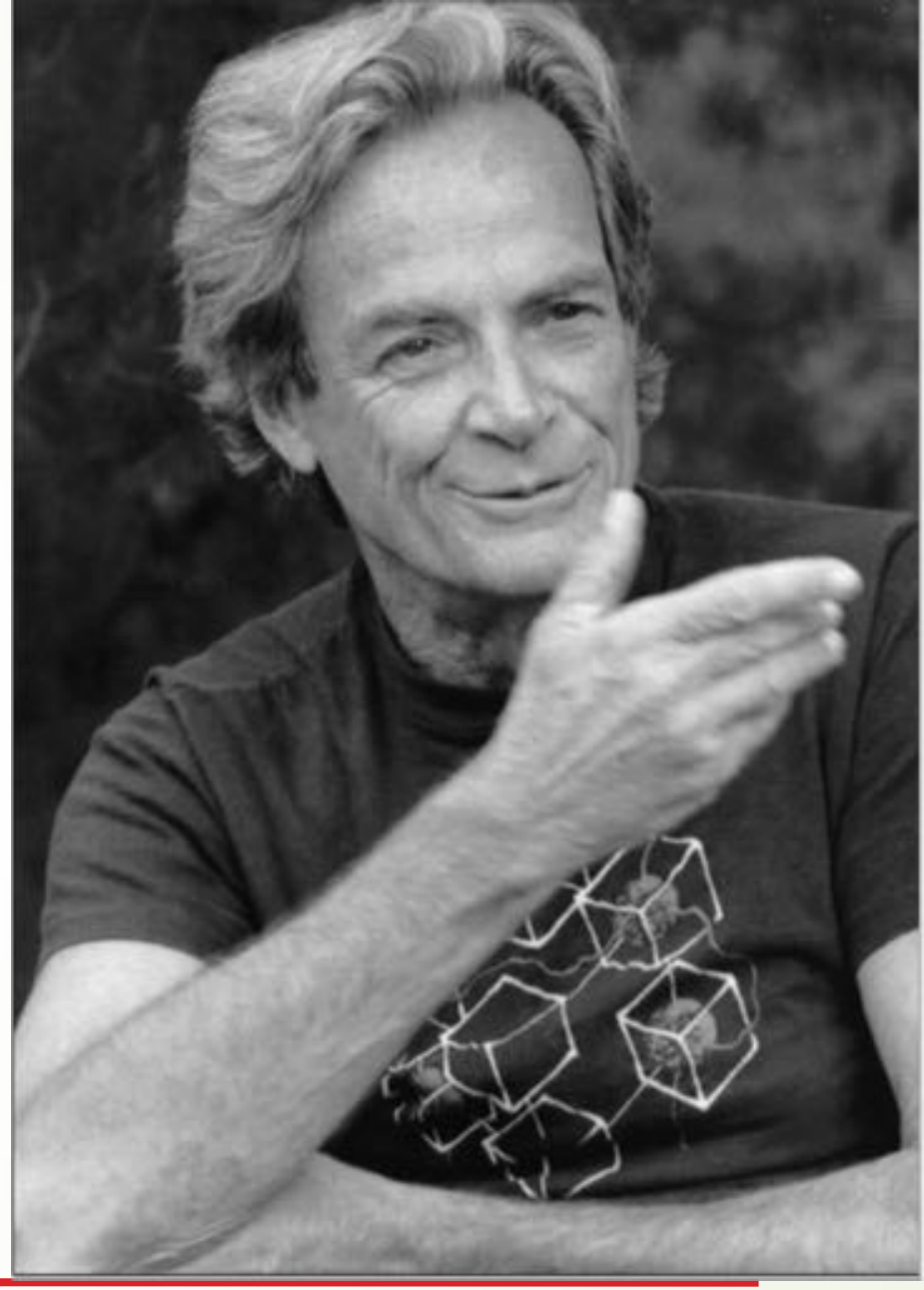
Mary Miller

MolSSI


TOP 10
STARTUP MISTAKES
#1
Building something nobody wants
Score: 300
36% of Tot.
#2
Hiring Poorly
Score: 153
18% of Tot.
#3
Lack of Focus
Score: 112
13% of Tot.
#4
Fail to execute Sales & Marketing
Score: 98
12% of Tot.
#5
#6
#7
#8
#9
#10
100
First Hits
www.100FirstHits.com
8. Spending Too Much Money 18 (2,1%)
9. Failing To Ask For Help 12 (1,4%)
10. Ignoring Social Media 6 (0,7%)
5. Not Having The Right Co-Founders 66 (7,9%)
6. Chasing Investors, Not Customers 45 (5,4%)
7. Not Making Sure You Have Enough Money 28 (3,3%)

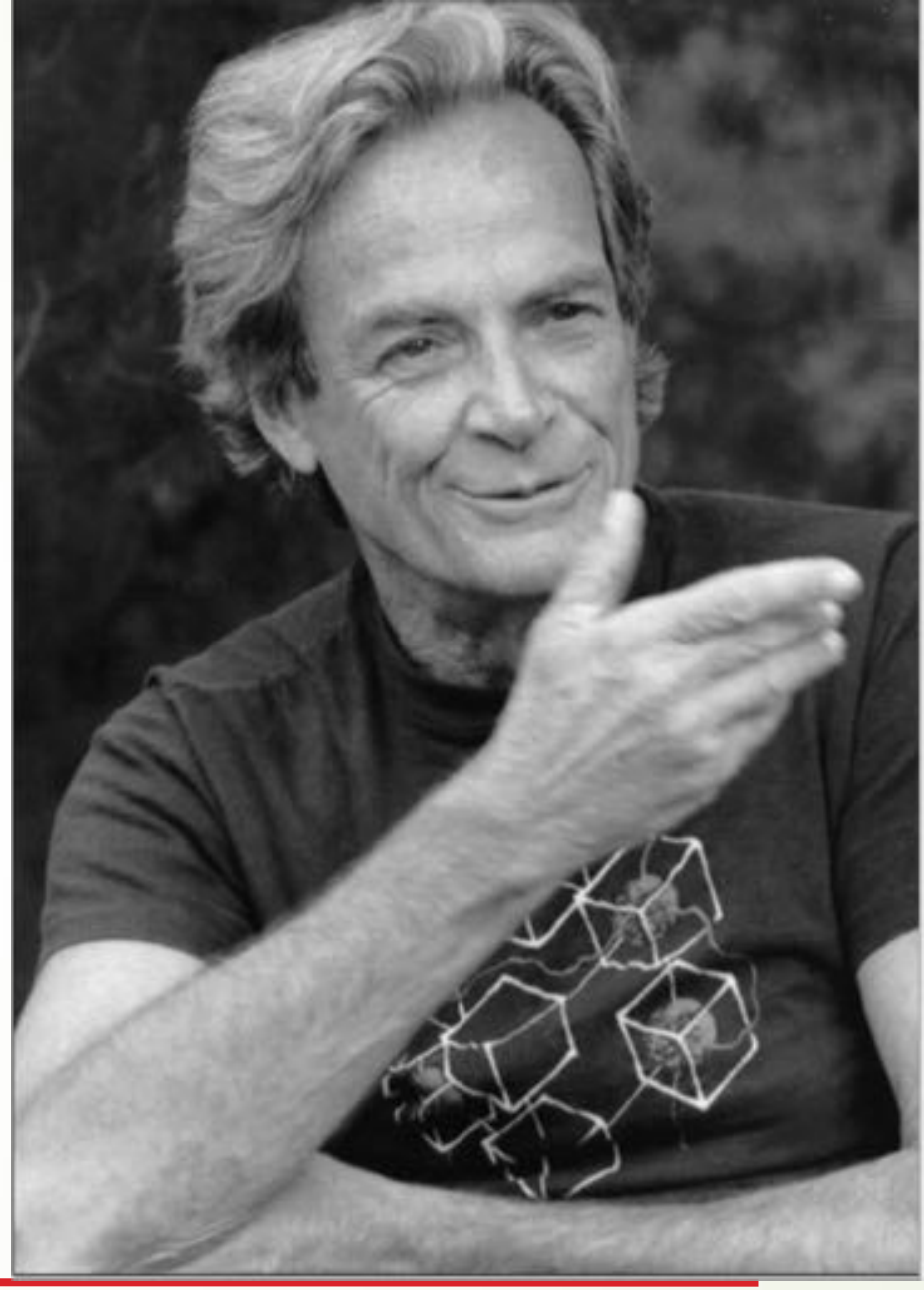
"The first principle is that you must not fool yourself

– Richard Feynman



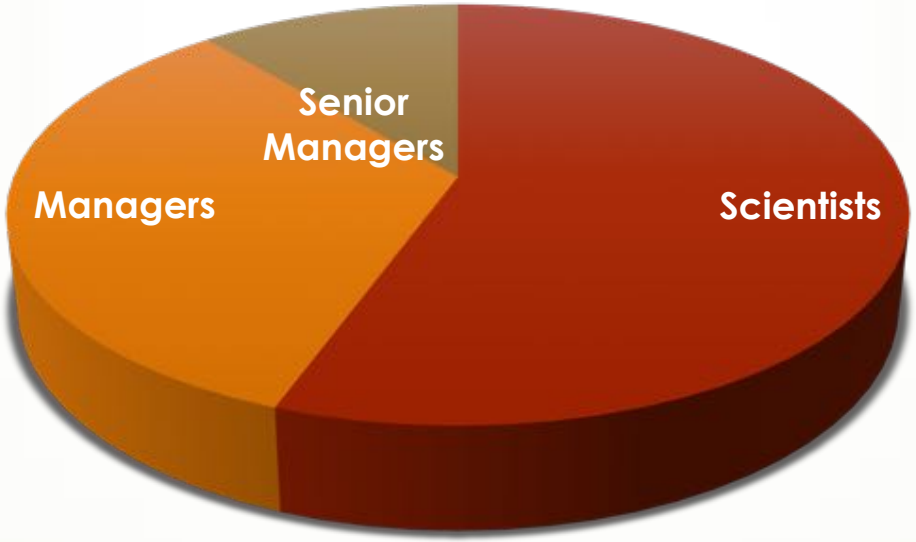
"The first principle is that you must not fool yourself, and you are the easiest person to fool."

– Richard Feynman



Who we talked to (so far…)

	Interview Count			
Total	27	5	16	6



What have we learned?

- Experimentalists are strongly skeptical of atomistic modeling

  The value of modeling must be demonstrated time and time again
- Time-to-solution is absolutely critical

  Solving the problem after the project has moved on is useless!
- Modeling experts shouldn't program

  Programming or scripting is a necessary evil
- Needed accuracy varies greatly depending on the problem (and time!)

  Need the entire range of tools at hand
- Many companies are trying to seamlessly integrate modeling & experiment

  One approach, not two!

MolSSI

Motivation for a Workflow Framework

Areas we Need to Improve

- Improved science
  - Reproducibility
  - Reducing errors
  - New tools and applications
  - Acknowledgement: citations
- Productivity
  - Automation
  - Ease-of-use
  - Efficient use of resources

Reproducibility

JCTC Journal of Chemical Theory and Computation
Article
pubs.acs.org/JCTC

Round Robin Study: Molecular Simulation of Thermodynamic Properties from Models with Internal Degrees of Freedom

Michael Schappals,[†] Andreas Mecklenfeld,[‡] Leif Kröger,[§] Vitalie Botan,[§] Andreas Köster,[∥] Simon Stephan,[†] Edder J. García,[†] Gabor Rutkai,[∥] Gabriele Raabe,[‡] Peter Klein,[⊥] Kai Leonhard,[§] Colin W. Glass,[#] Johannes Lenhard,[▽] Jadran Vrabec,[∥] and Hans Hasse*[,†]

J. Chem. Theory Comput., **2017**, *13* (9), pp 4270–4280
DOI: 10.1021/acs.jctc.7b00489

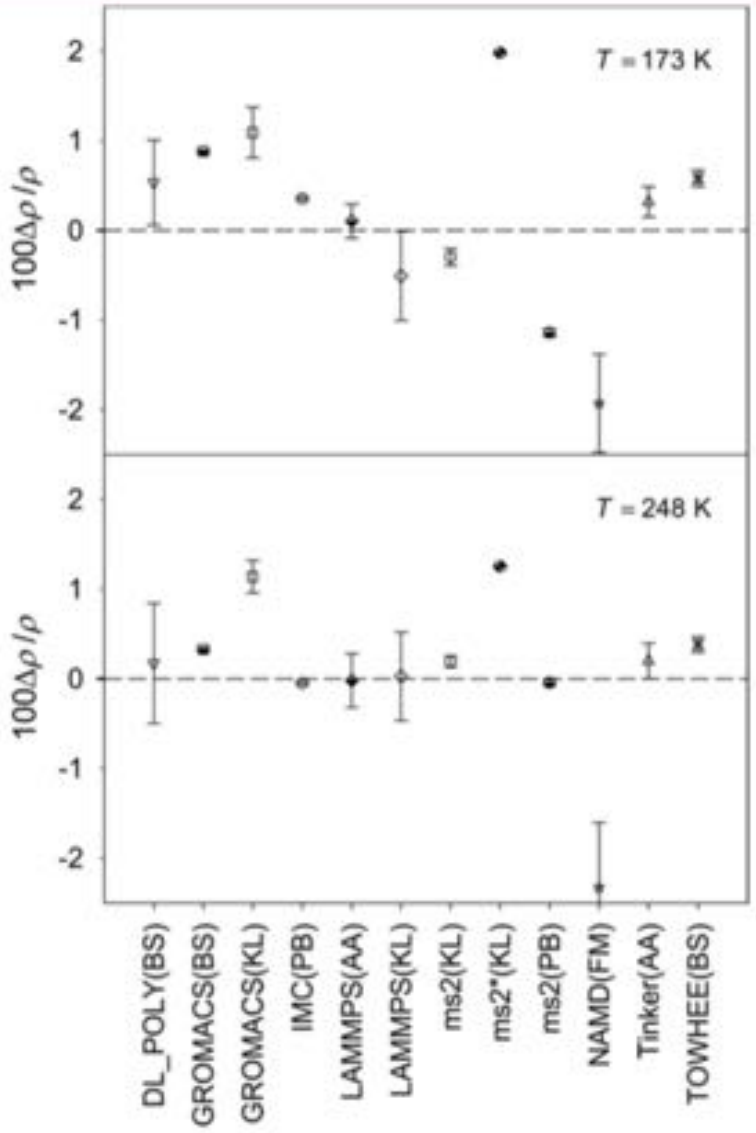


Figure 10. Statistical uncertainty of the data obtained for the density of *n*-butane at 41 MPa and 173 (top) and 248 K (bottom) from the OPLSAMBER force field. Symbols: mean values with error bars determined from block averages of the production phase. Dashed line: arithmetic mean of all results.

What is Reproducibility?

- Ability to mechanically reproduce a given simulation ✓
- Ability for someone else to recreate a simulation ?
- Ability for someone else to reproduce, then change, a simulation ??

Ease-of-Use

Gaussian 76 input "deck"

Column Guide

```
                00000000011111111112222222222333333333344444444445555555555
                12345678901234567890123456789012345678901234567890123456789
CARD   1:  S   1 1
CARD   2:          WATER MOLECULE   RHF/STO-3G
CARD   3:   0 1
CARD   4:    8
CARD   5:    1    1 0.96
CARD   6:    1    1 0.96      2 109.471
CARD   7:
                12345678901234567890123456789012345678901234567890123456789
                00000000011111111112222222222333333333344444444445555555555
```

Gaussian 8X (or 16) Input File

```
# HF/STO-3G(d)
water energy
0   1
O
H  1  0.96
H   1 0.96  2 109.471
```

```
                 00000000011111111112222222222333333333344444444445555555555
                 12345678901234567890123456789012345678901234567890123456789
CARD   1:        S   1 1
CARD   2:                WATER MOLECULE   RHF/STO-3G
CARD   3:         0 1
CARD   4:         8
CARD   5:         1     1 0.96
CARD   6:         1     1 0.96       2 109.471
CARD   7:
                 12345678901234567890123456789012345678901234567890123456789
                 00000000011111111112222222222333333333344444444445555555555
```

Fixed format!

```
# HF/STO-3G(d)
water energy
0  1
O
H  1  0.96
H  1 0.96  2 109.471
```

Keywords rather than numbers at positions

Free format!

Element symbols instead of atomic numbers

GAMESS 2005 … or 2016

```
$CONTRL SCFTYP=RHF RUNTYP=ENERGY $END
$BASIS GBASIS=STO NGAUSS=3 $END
$DATA
STO-3G TEST CASE FOR WATER
Cnv 2
Oxygen 8.0 0.0 0.0 0.0
Hydrogen 1.0 -0.758 0.0 0.545
$END
```

GAMESS has >1000 keywords!

LAMMPS – Molecular Dynamics

```
# Rhodopsin model

units            real
neigh_modify     delay 5 every 1

atom_style       full
bond_style       harmonic
angle_style      charmm
dihedral_style   charmm
improper_style   harmonic
pair_style       lj/charmm/coul/long &
                   8.0 10.0
pair_modify      mix arithmetic
kspace_style     pppm 1e-4
```

```
read_data        data.rhodo

fix              1 all shake 0.0001 5 0 m 1.0 a 232
fix              2 all npt temp 300.0 300.0 100.0 &
                 z 0.0 0.0 1000.0 mtk no pchain 0 &
                 tchain 1

special_bonds    charmm

thermo           50
thermo_style     multi
timestep         2.0

run              100
```

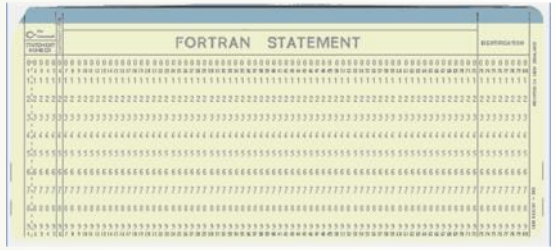
```
LAMMPS data file from restart file: timestep = 5000,
procs = 1

32000 atoms
27723 bonds
40467 angles
56829 dihedrals
1034 impropers

68 atom types
115 bond types
243 angle types
453 dihedral types
19 improper types

-27.5 27.5 xlo xhi
-38.5 38.5 ylo yhi
-36.3646 36.3615 zlo zhi

Masses

1 1.008
2 1.008
3 1.008
4 1.008
5 1.008
6 1.008
…
```

191,072 lines!

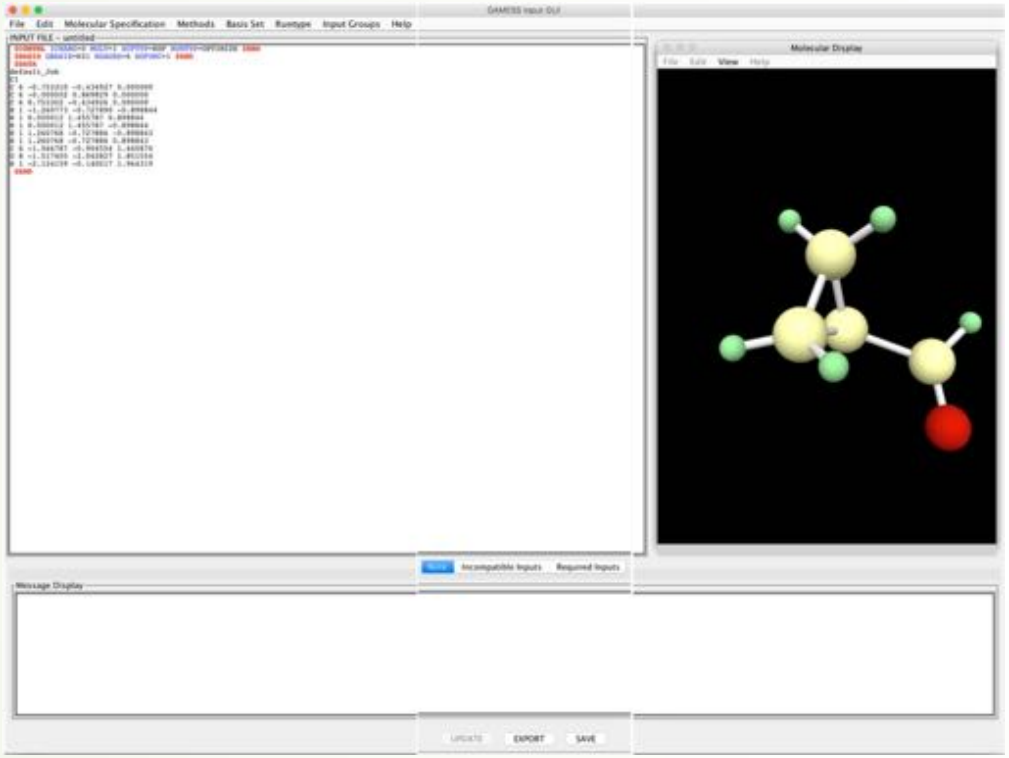
6.3 MB

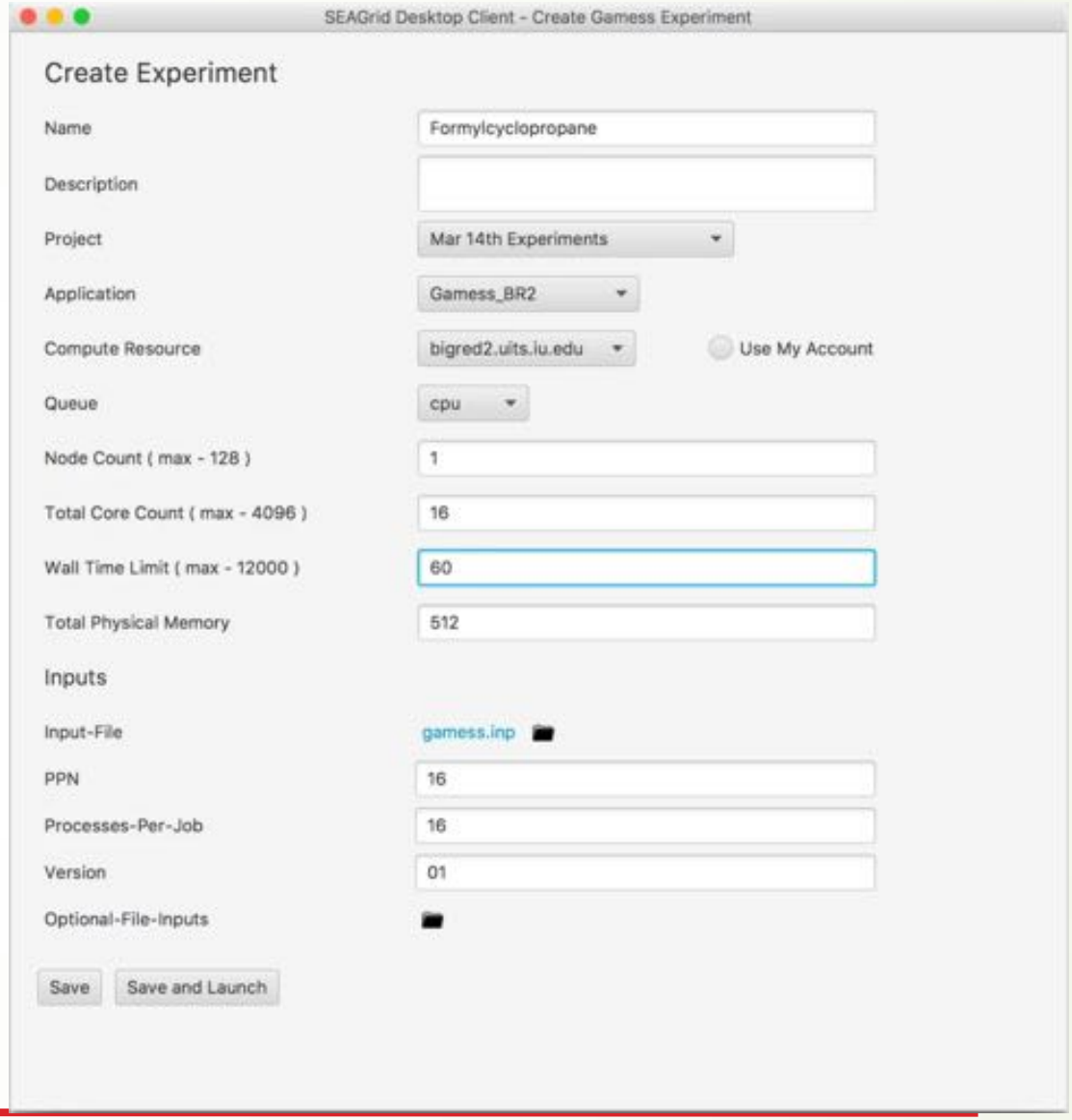
```
Angle Coeffs

1 52 112.3 0 0
2 50 112 0 0
3 50 108.2 0 0
4 52 108 0 0
5 52 108 0 0
6 52 108 0 0
7 50 109.5 0 0
8 50 107 0 0
…
237 145 108 0 0
238 120 120 0 0
239 98.9 111.6 0 0
240 90 125.9 160 2.2576
241 90 125.9 160 2.2576
242 100 124 70 2.225
243 80 104.3 0 0


Dihedral Coeffs


1 0.14 3 0 1
…
```

We can do better!



SEAGRID





https://seagrid.org/home

Possibility: Python UI, à la pymatgen

```
>>> lattice = mg.Lattice.cubic(4.2)
>>> structure = mg.Structure(lattice, ["Cs", "Cl"],
...                          [[0, 0, 0], [0.5, 0.5, 0.5]])
>>> structure.volume
74.088000000000008
>>> structure[0]
PeriodicSite: Cs (0.0000, 0.0000, 0.0000) [0.0000, 0.0000, 0.0000]
>>>
>>> # You can create a Structure using spacegroup symmetry as well.
>>> li2o = mg.Structure.from_spacegroup("Fm-3m", mg.Lattice.cubic(3),
["Li", "O"],
[[0.25, 0.25, 0.25], [0, 0, 0]])
>>>
>>> # Integrated symmetry analysis tools from spglib.
>>> from pymatgen.symmetry.analyzer import SpacegroupAnalyzer
>>> finder = SpacegroupAnalyzer(structure)
>>> finder.get_spacegroup_symbol()
'Pm-3m'
>>>
>>> # Convenient IO to various formats. You can specify various formats.
>>> # Without a filename, a string is returned. Otherwise,
>>> # the output is written to the file. If only the filenmae is provided,
>>> # the format is intelligently determined from a file.
>>> structure.to(fmt="poscar")
>>> structure.to(filename="POSCAR")
>>> structure.to(filename="CsCl.cif")
>>>
>>> # Reading a structure is similarly easy.
>>> structure = mg.Structure.from_str(open("CsCl.cif").read(), fmt="cif")
>>> structure = mg.Structure.from_file("CsCl.cif")
```

…or Atomic Simulation Environment (ASE)

```
>>> # Example: structure optimization of hydrogen molecule
>>> from ase import Atoms
>>> from ase.optimize import BFGS
>>> from ase.calculators.nwchem import NWChem
>>> from ase.io import write
>>> h2 = Atoms('H2',
...            positions=[[0, 0, 0],
...                       [0, 0, 0.7]])
>>> h2.calc = NWChem(xc='PBE')
>>> opt = BFGS(h2)
>>> opt.run(fmax=0.02)
BFGS:   0  19:10:49    -31.435229     2.2691
BFGS:   1  19:10:50    -31.490773     0.3740
BFGS:   2  19:10:50    -31.492791     0.0630
BFGS:   3  19:10:51    -31.492848     0.0023
>>> write('H2.xyz', h2)
>>> h2.get_potential_energy()
-31.492847800329216
```

…or Jupyter Notebooks?

In [3]:
```
from chemview import MolecularViewer

import numpy as np

def parse_mol_string(string):
    lines = string.splitlines()
    # lines 0-2 are header/comments

    # line 3 is counting
    natoms = int(lines[3][0:3])
    nbonds = int(lines[3][3:6])

    coords = []
    types = []
    bonds = []
    bond_types = []

    for i in range(natoms):
        at_fields = lines[i + 4].split()
        x, y, z, typ = at_fields[:4]
        coords.append([float(x), float(y), float(z)])
        types.append(typ)

    offset = natoms + 4
    for i in range(nbonds):
        s = lines[offset + i][0:3]
        e = lines[offset + i][3:6]
        t = lines[offset + i][6:9]
        bonds.append((int(s),int(e)))
        bond_types.append(int(t))

    return np.array(coords)/10, np.array(types), np.array(bonds) - 1
```

In [4]:
```
import urllib.request

response = urllib.request.urlopen('https://cactus.nci.nih.gov/chemical/structure/caffeine/sdf')
textData = response.read().decode('utf-8')
coords, types, bonds = parse_mol_string(textData)

mv = MolecularViewer(coords, {'atom_types': types, 'bonds': bonds}, width=300, height=300)
mv.ball_and_sticks()
mv
```

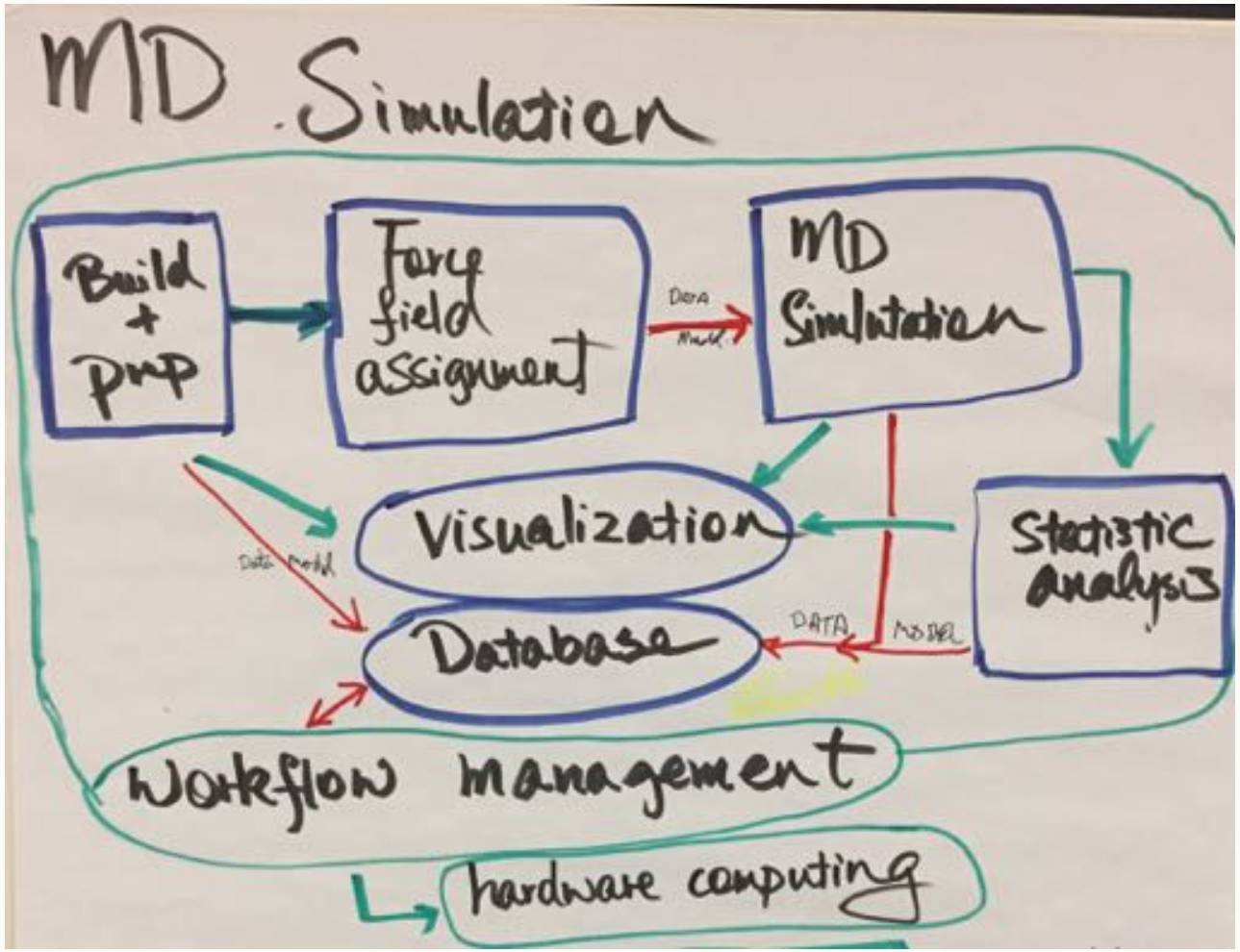
```
/home/herman/miniconda3/envs/chemview/lib/python3.5/site-packages/traitlets/traitlets.py:565: FutureWarning: c
omparison to `None` will result in an elementwise object comparison in the future.
  silent = bool(old_value == new_value)
```

Approaches

- Gateway / Portal
  - Dialogs and perhaps 3-D graphics (GUI)
  - No programming
  - Can handle all types of reproducibility
  - Good GUIs are difficult work
  - Concern that is "black box"
- Programming API
  - An API for Python or similar
  - Requires reasonable programming skills
  - May handle some reproducibility
  - Good documentation is difficult work
  - Concern over too much flexibility (where do I begin?)

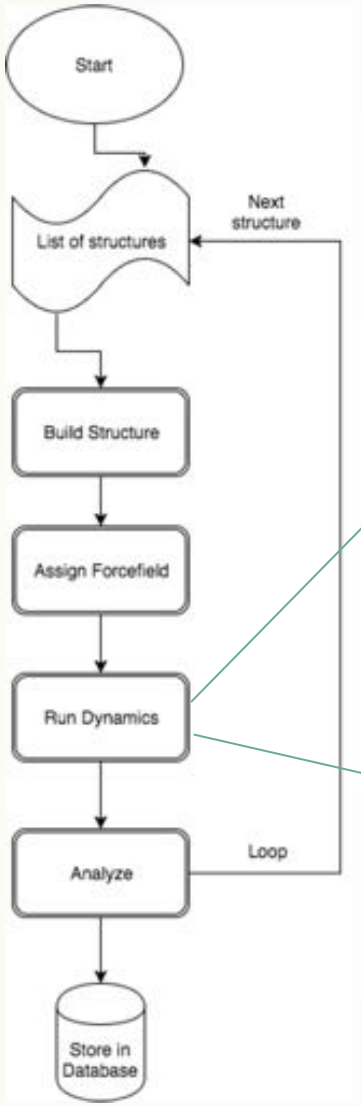
MolSSI

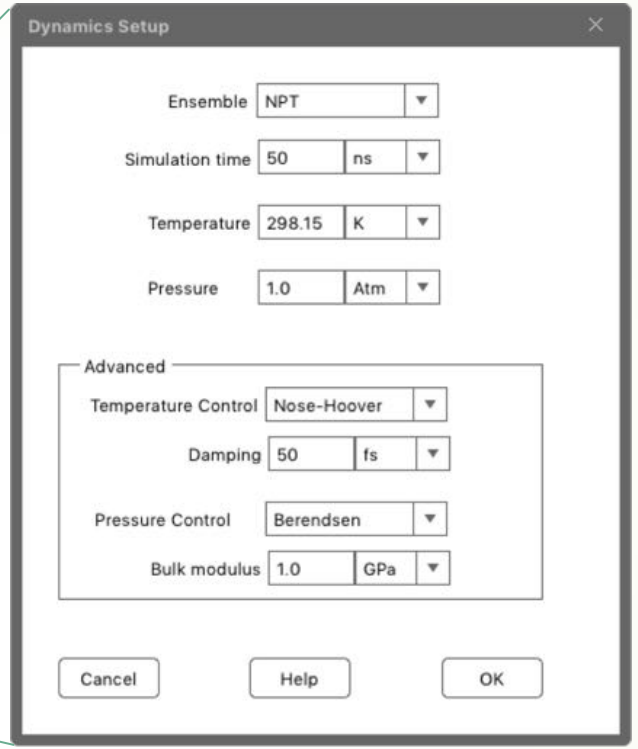
How do we put this on the computer?



Like this?

Editable flowchart

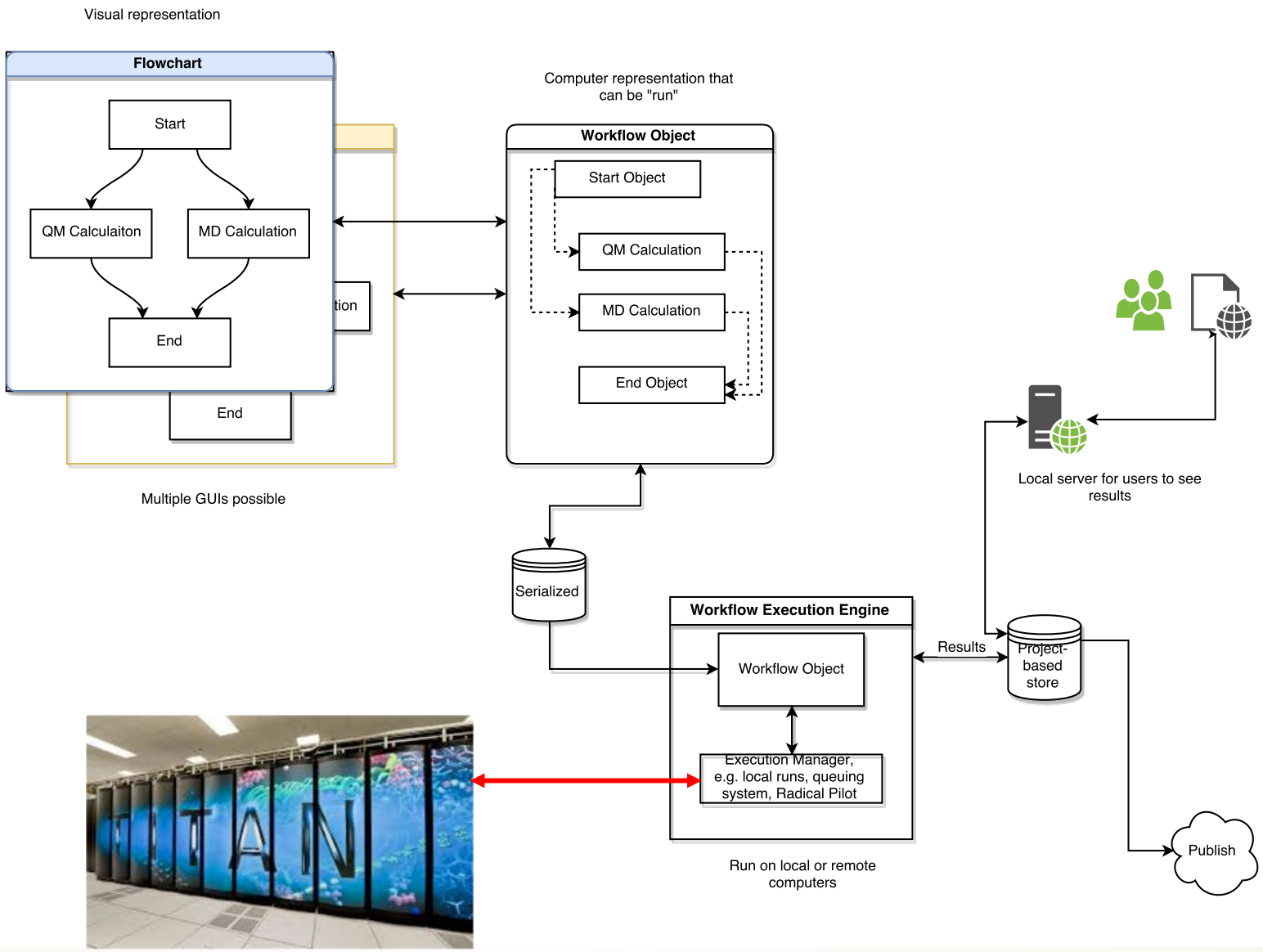




Dialog for setting parameters in node

Workflow Framework

MolSSI Workflow Framework



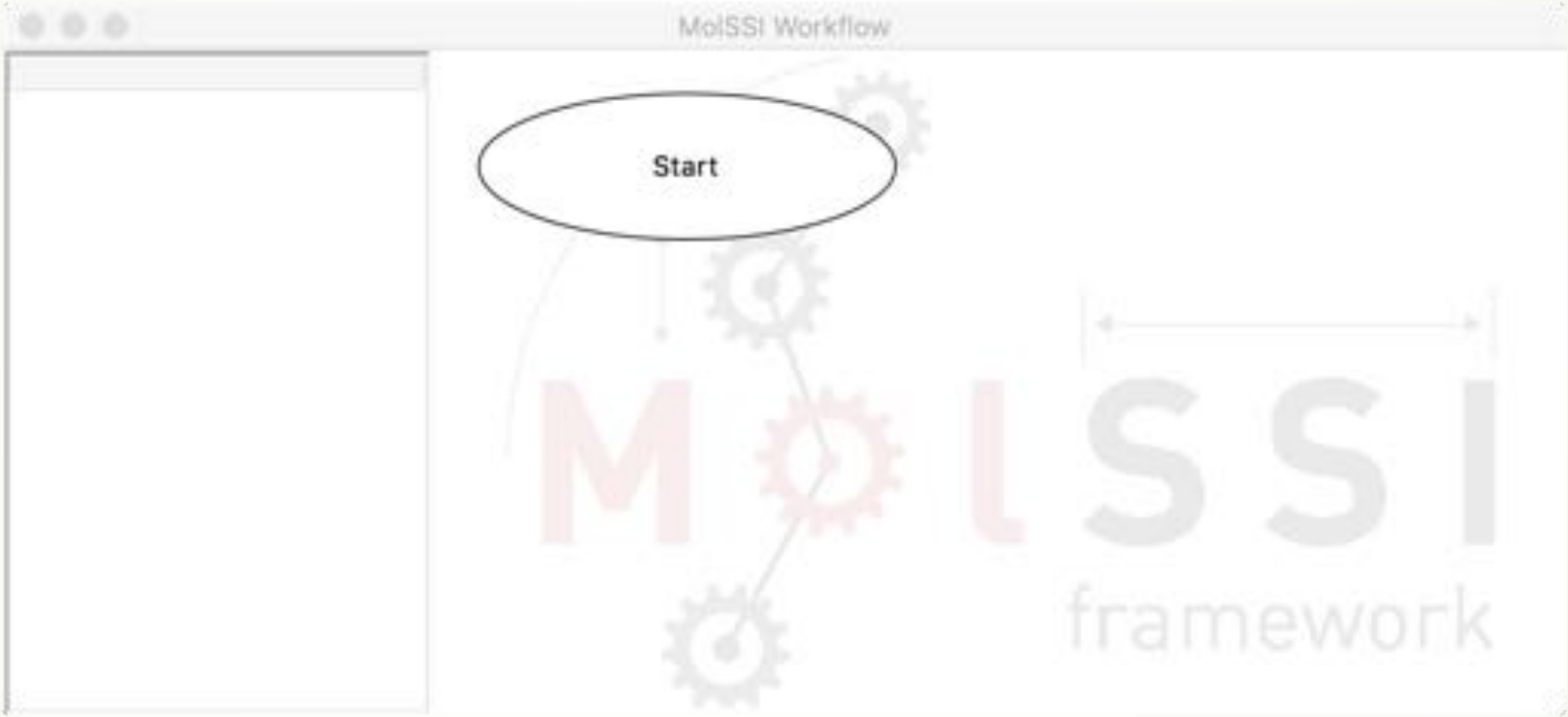
What does the user require?

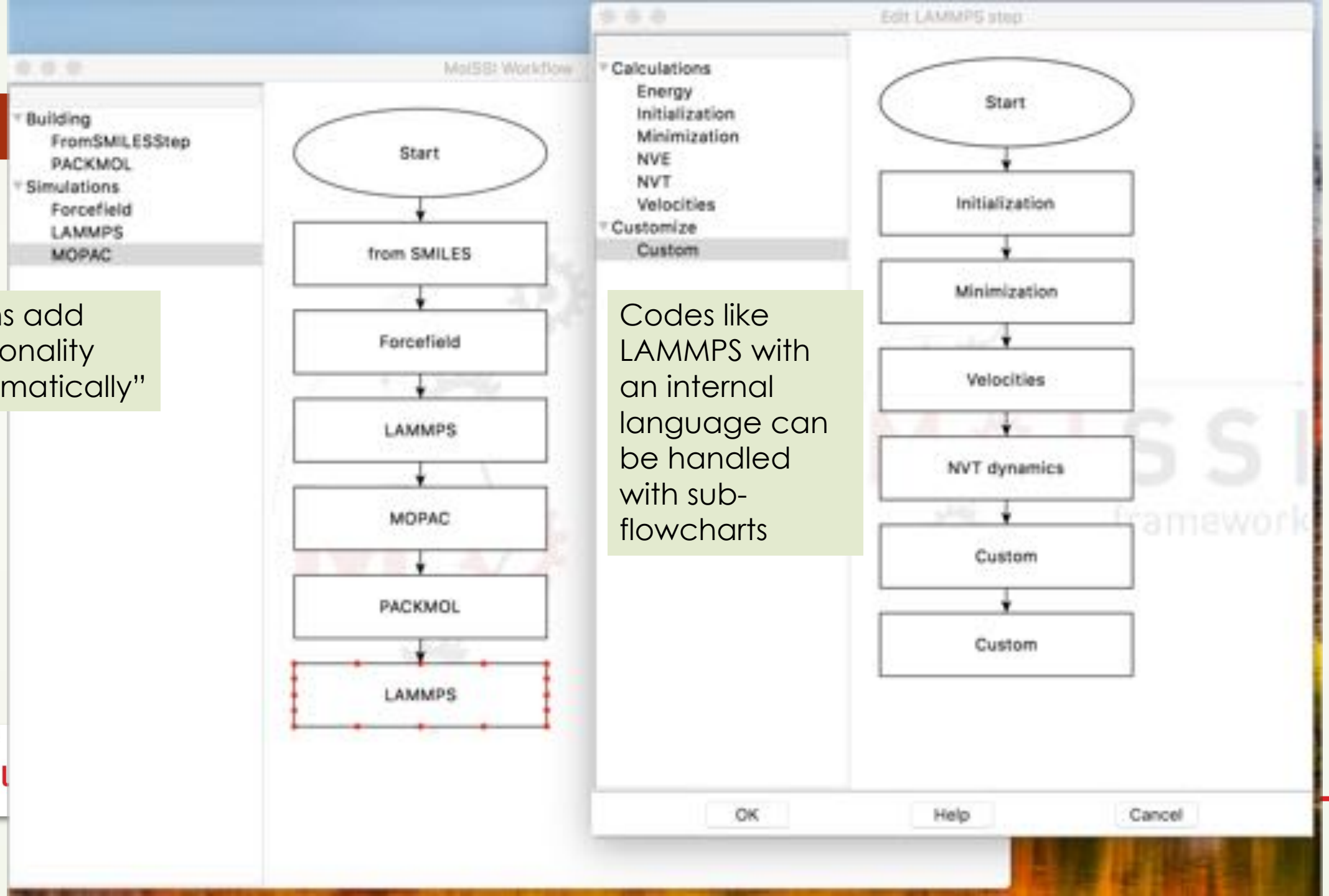
- Complete environment
  - Builders, editors, database retrieval etc. for model preparation
  - Wide range of simulation tools
  - Analysis tools
  - Saving results to files, databases
  - Creating graphs and other visual representations such as movies
  - Control structures, decision handling, error capture, etc.
- Seamless integration with computing resources
- Easy to use, learn, install and manage
- Publishing, including proper citations
- Shareable protocols
- Reproducibility

What does the framework itself require?

- Application agnostic … but "knows" chemistry
- Long lasting – certainly 15 years, preferably longer.
  - As light as possible
  - As simple as possible, with extensible APIs
- Support for multiple underlying computational "workflow" management systems
- Dispersed development of application portion (plugins?)
- Large developer community
- Larger user community!

Empty Framework





Plugins add functionality "automatically"

Codes like LAMMPS with an internal language can be handled with sub-flowcharts

The Framework Provides

- Mechanism for plugging in modules
- Containers for the GUI
- Commonly used data structures (workflow, molecule,…)
- Connections to the database & computational resources
- Well-defined API with utility libraries
- Saving and restoring workflows
- Citation manager

… almost nothing that a user sees!

Must not change often or much!

Other groups provide plugins

- (Almost) completely independent of each other
- Are responsible for everything in their plugin!
- There can be multiple different plugins for a code
- There can be multiple codes fronted by one plugin
- Doesn't have to wrap a code (if the task is quick)
- Can be developed by anyone, does not have to be the simulation code developers.

Recap: Areas to Improve

- Improved science
  - Reproducibility ✓
  - Reducing errors ✓
  - New tools and applications (yes

    Makes it easier to mix, match and combine. Quite complex simulations can be captured in workflows, though adding translators and tools (plugins) will be needed sometimes.)
  - Acknowledgement: citations ✓
- Productivity
  - Automation ✓
  - Ease-of-use ✓
  - Efficient use of resources (yes

    Since all simulations funnel through one submission section, more opportunities to pick computers, set tuning parameters.)

MolSSI

Summary

- Neutral framework for atomistic simulations
- Uses plugins to decentralize
- Provides central concept of a “system”
- Multiple frontends supports (application, web portal,…)
- Provides support libraries – cheminformatics, statistics, graphing, …
- Provides citation manager (but plugins have to do their part!)
- Hides complexity of job submission
- Stores all results in a personal or group local datastore
- Open source

You can help!

I want to create a community to guide this!

I haven't figured out the best way – suggestions?
Email? Slack? Google Docs?

So, for the moment, email me to be included

psaxe@vt.edu