# Lessons learned from SAM Simulation Core application development

Joseph Ranalli

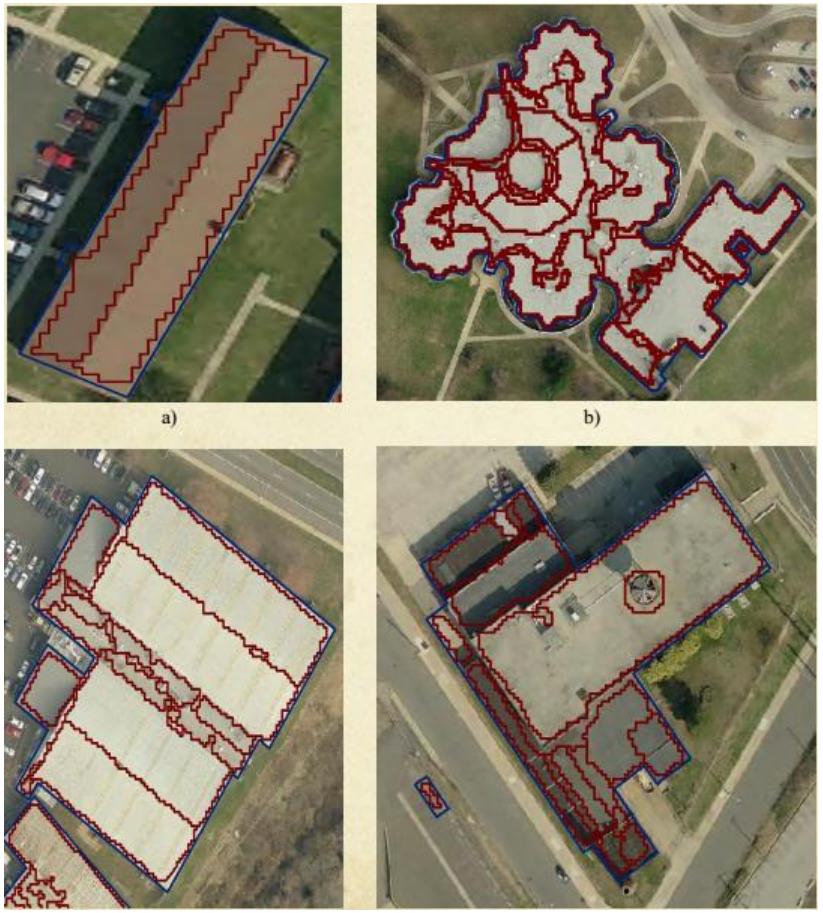
Penn State Hazleton

# Research Team

2 campuses, 3 departments, 4 countries, many disciplines

# PVAnalyst 2.0

- ArcGIS plus System Advisor Model (SAM)
  - Wrapper code to make SAM callable (Open Source)
  - Integration code for ArcGIS to make calls
- LiDAR data provides roof slope/azimuth and dimensions for City of Philadelphia.
- Calculations used to simulate a solar system on each roof



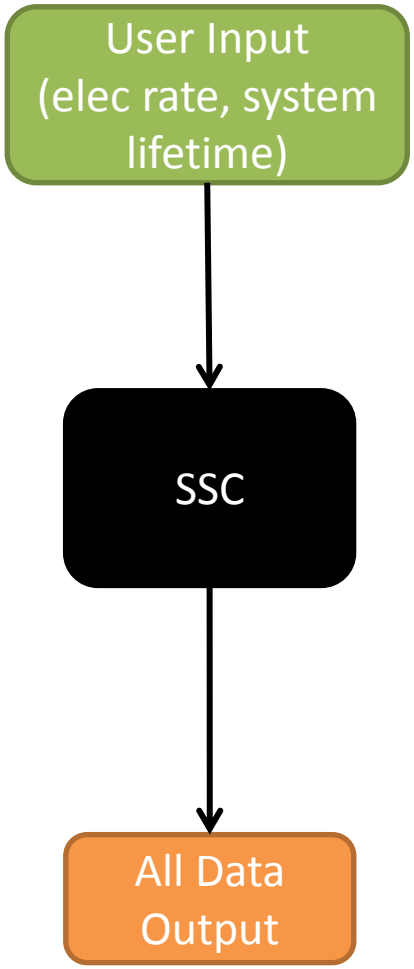
# SAM Simulation Core (SSC)

- Development kit for integrating SAM calculations in software
  - Matlab, Java, Python, c, c#
  - Examples provided
- Access to most SAM model functionality available
- Several SAM calculations must be done manually
  - e.g. combined cost of components to system cost



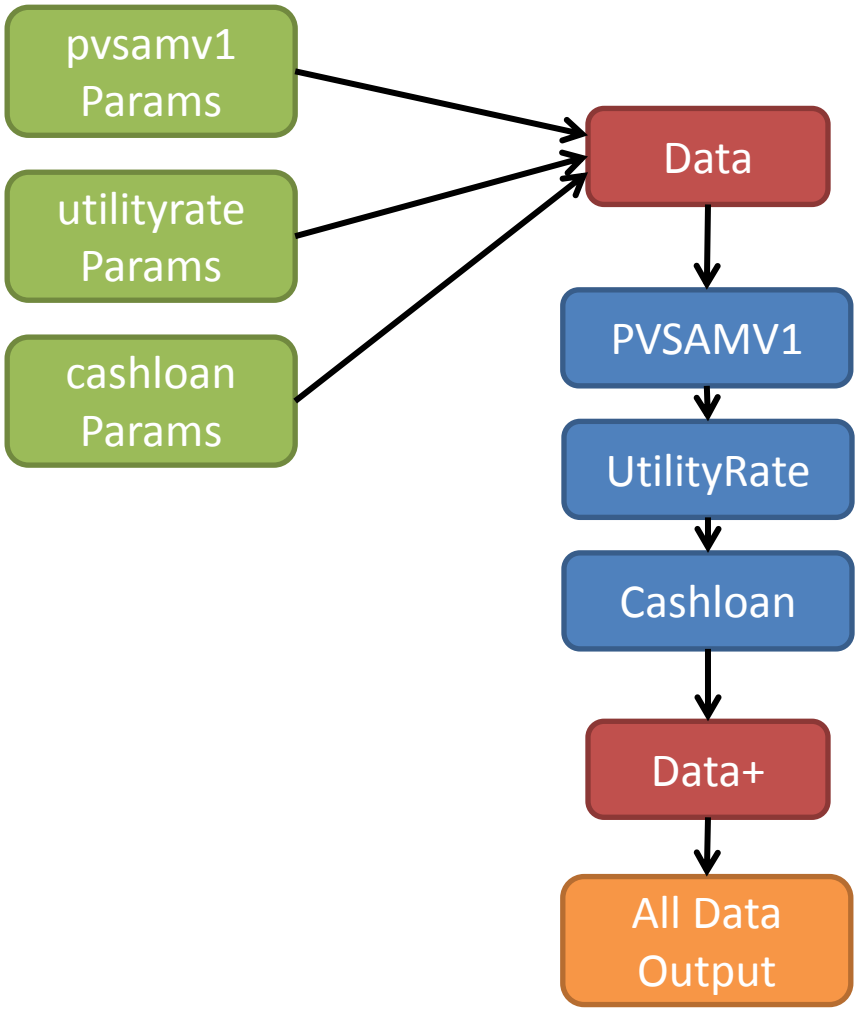

# Ideal Client Code Interface

- Black box operation
- Could work with SSC examples, but challenges exist
  - Example calculation contains 468 parameters!
    - Inconvenient to specify these within code
    - Uses strings to ID parameters
  - Coupling between modules makes testing difficult



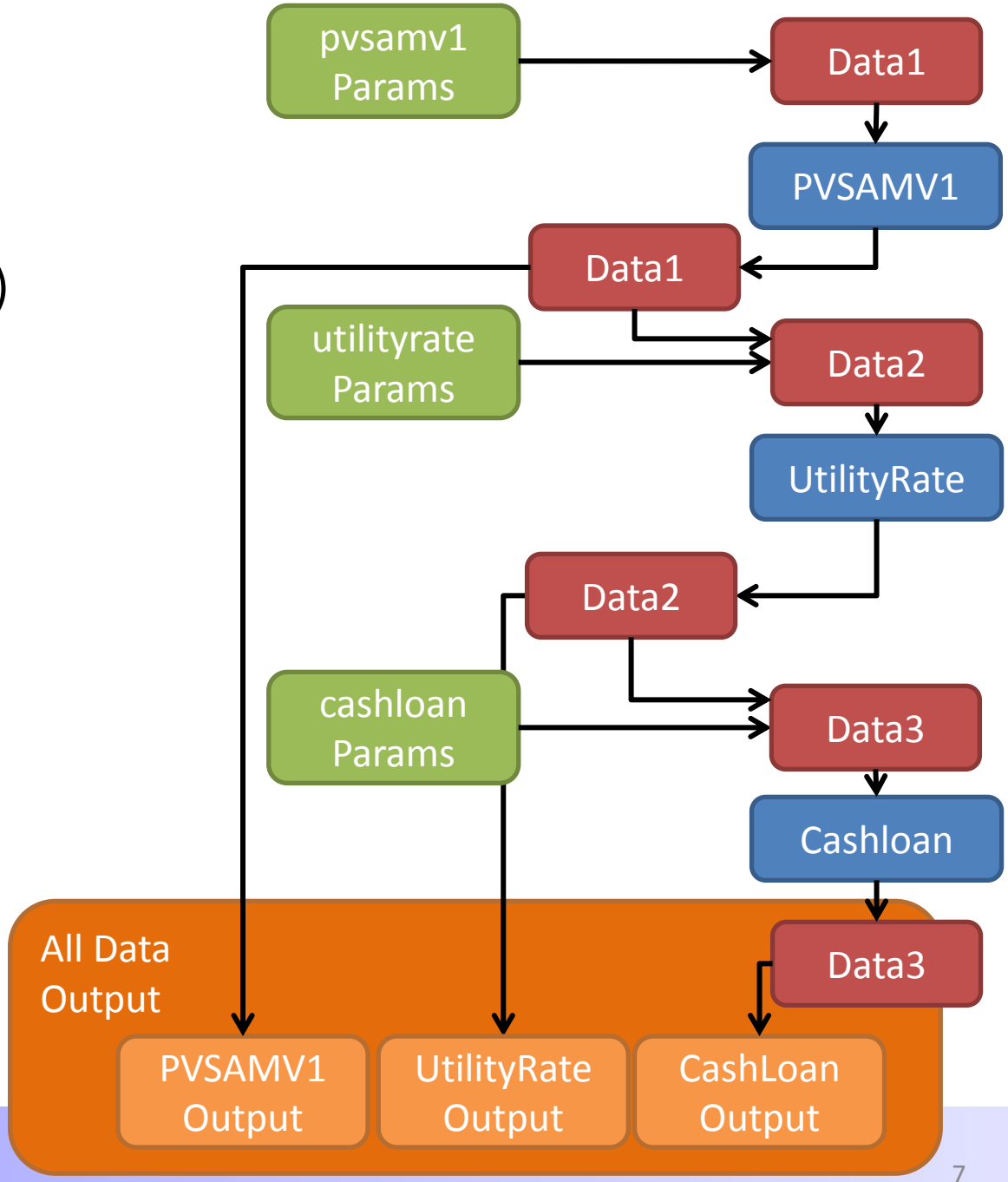
# SSC Example Data Flow

- All parameters values set within DATA object
- Output from PVSAM1 within DATA as it enters UTILITYRATE
- Input to UTILITYRATE depends on both UTILITYRATE params and PVSAM1 outputs
- Output DATA object contains all results in one structure
- Coupling between each step
  - Difficult to separate effect of each module



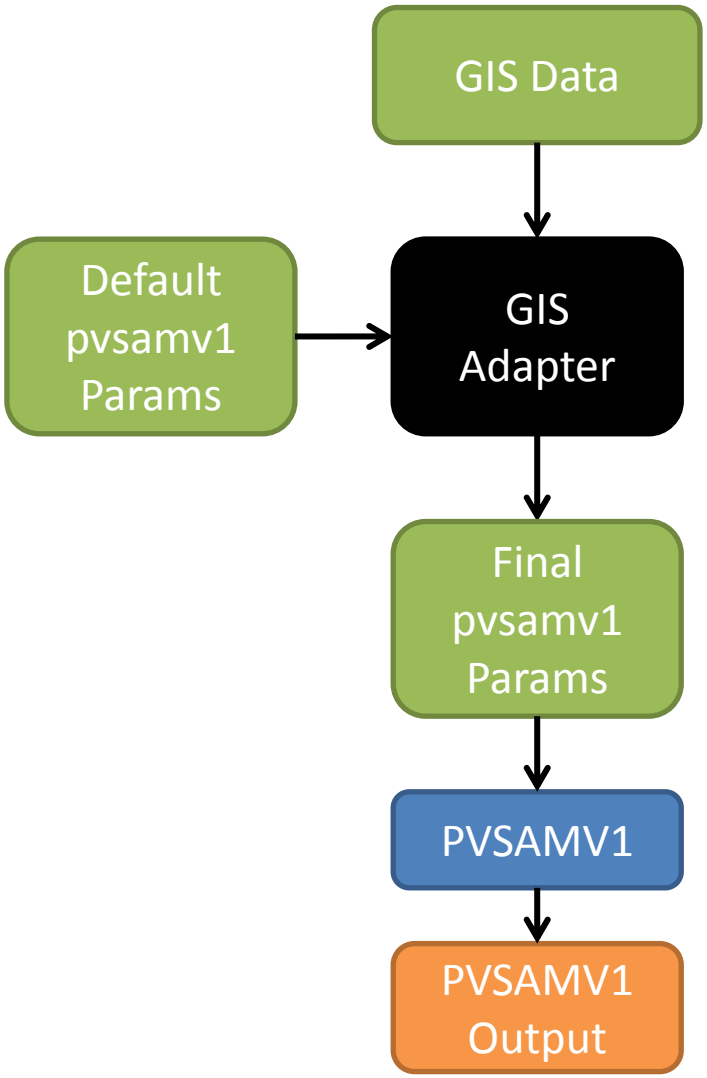
# SSC Wrapper

- Form complete set of input parameters (including prev. outputs)
- Use new DATA object as input to next module
- Wrap multiple outputs into single object
- More complex, but has advantages over default:
  - Explicitly decouples modules
  - Functions of each module can be investigated independently



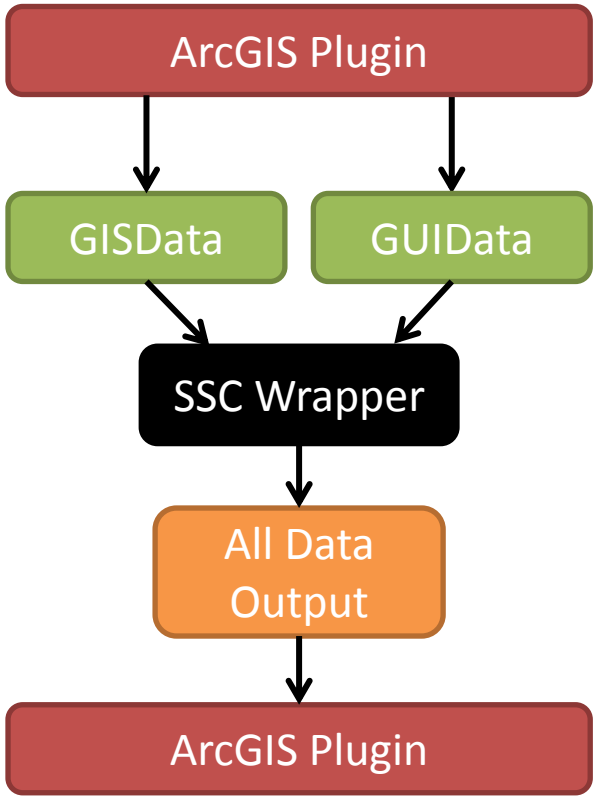
# Increased complexity, more flexibility

- Write custom adapters
  - Translate user Inputs into SSC parameter sets
- Parameters used to run Modules
  - Outputs compiled at the end
- Mimics “script” coding based on user data objects



# Interface with ArcGIS

- Create GISData and GUIData objects within ArcGIS
- Pass simulation request to wrapped SSC library
- Obtain desired data from the output
  - System Size
  - Solar Savings
  - LCOE
  - Any other SAM outputs



# Pseudocode Example PVSAMV1 Run

```
DataOutput Run(GISData gis, GUIData gui){

  // Initialize and Apply GISData Values
  PVSAMV1Settings settings = PVSAMV1Settings.getDefault();
  GISAdapter.applySettings(settings,gis);

  // Inverter settings
  float arraypow = settings.Nmodules*settings.ModulePower;
  InvrtSet inv = new InvrtSet("default", arraypow * 1.15);
  settings.inverter_model = inverter;

  // Execute PVSAMV1
  PVSAMV1Output pvout = ModuleRunner.runModule(settings);

  // Continue for UtilityRate and CashLoan modules …
}
```

# Pseudocode Example GISAdapter

```
public void applySettings(PVSAMV1Settings stgs, GISData gis){

   // Set array orientation flat on the roof
   pvss.subarray1_tilt = gis.tilt;
   pvss.subarray1_azimuth = gis.azimuth;

   // Calculate array layout based on GIS roof data
   int Nmods = calcRoofFit(gis.width, gis.height);
   int nSer = calcNSer(Nmods,stgs.invVmax,stgs.modVmax);
   int nPar = calcNPar(Nmodules,stgs.invVMax,stgs.modVmax);

   // Translate to the SSC settings
   pvss.modules_per_string = nSer;
   pvss.strings_in_parallel = nPar;

}
```

# Work-in-Progress Lessons Learned

- SSC can be adapted to operate conveniently within an application
- Primary challenges are number of inputs to SSC modules, and use of string identifiers
- Tradeoffs must be made between ease-of-use and level of user control detail
- Community feedback and interest could lead to a universal library wrapper for SSC for application development

# Thank You!

# Questions?


Contact Info:

Joe Ranalli

jranalli@psu.edu