Software Engineering for the SimVascular Project

David Parker, PhD
Research Software Engineer
Stanford Research Computing Center
Marsden Cardiovascular Biomechanics Computation Laboratory
Stanford University

Stanford MEDICINE

STANFORD BIOENGINEERING

Lucile Packard Children's Hospital AT STANFORD

Background

Stanford Research Computing Center

Manage and support more than
- 35 petabytes of data
- 2,000 HPC servers
- 40,000 CPU cores, 2,040 GPUs
- 13,000 user accounts

20 team members (two research software engineers)
- Help researchers transition their analyses and models from the desktop to HPC systems
- Manage a variety of compute clusters for various Stanford schools, departments, and/or labs

My background
- BS Computer Science (GaTech), PhD Biomechanics (Stanford)
- Visualization, Computational mechanics, Scientific computing, Biomechanics, Biophysics, Bioinformatics, Geometric computation, Structural biology
- Industry: Centric Engineering, Autodesk, Thermo Fisher, …
- Academia: GaTech, Stanford (Vascular Surgery, SimBios)

Background

Marsden Research Group

Develops fundamental computational methods for the study of

- Cardiovascular disease progression
- Surgical methods
- Treatment planning and medical devices

Focus on patient-specific modeling in pediatric and congenital heart disease, as well as adult cardiovascular disease

Develops open-source modeling and simulation tools to enable clinical and basic science research

- **SimVascular** - Interactive image-based modeling (C++, Python)
- **svSolver** - Computational fluid dynamics solver (Fortran, C++)
- **svFSI** - Fluid-structure interaction solver (Fortran)
- **svOneDSolver** - Flow in deformable 1D hemodynamic networks (C++, Python)

SimVascular

Open-source image-based modeling application

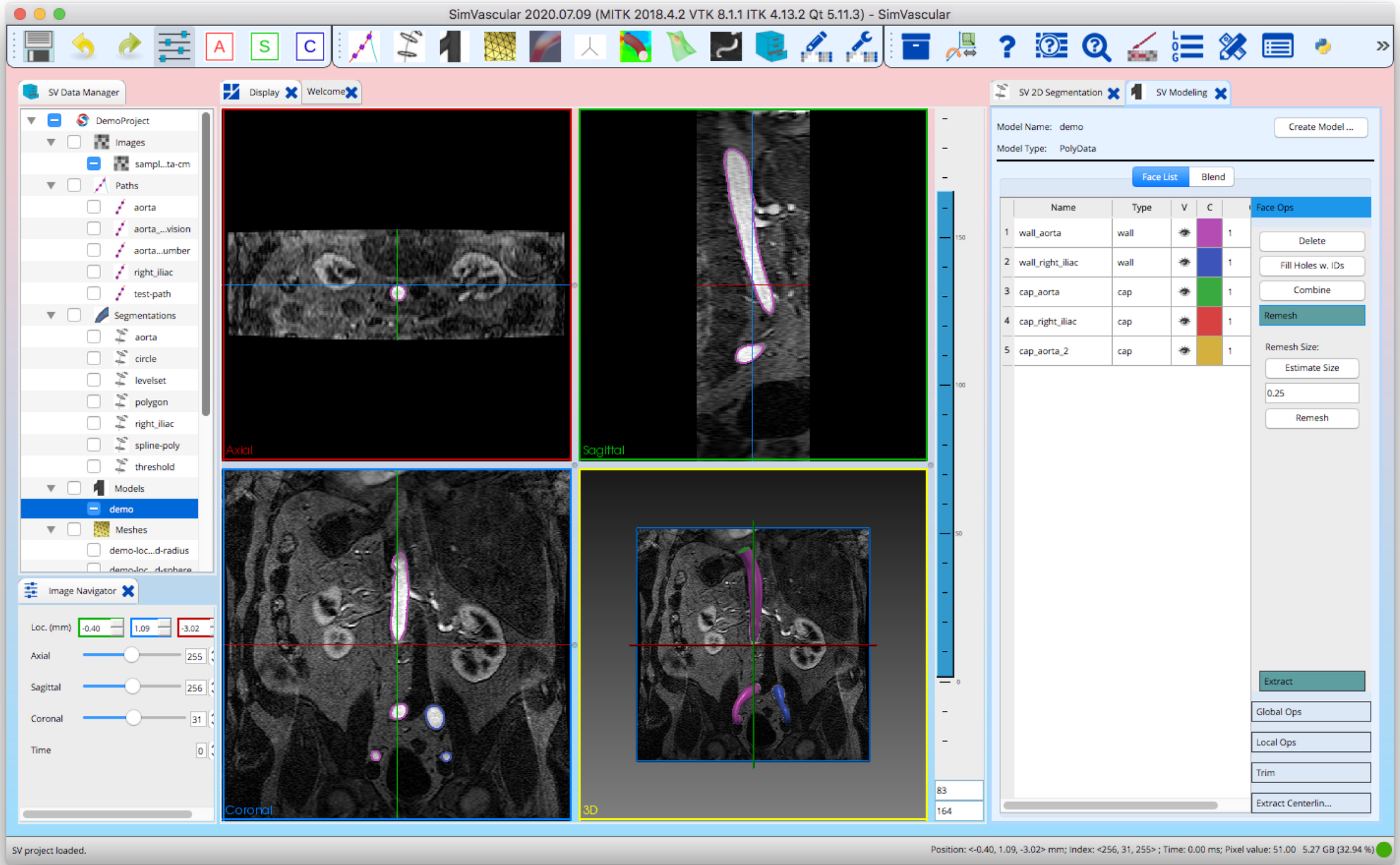
- Initial release in 2007
- Rewritten in 2013 with NSF support
- Hosted on GitHub (https://github.com/SimVascular/SimVascular)
- 1,700 user and 4,386 unique downloads
- 100 citations in publications/abstracts
- Used in coursework for project-based learning
- Download installers from SimTK for MacOS, Ubuntu and Windows

Web-based documentation (http://simvascular.github.io/)

- Tutorial-based guides
- No user manual

User forum hosted on SimTK

SimVascular



SimVascular Modeler

SimVascular software

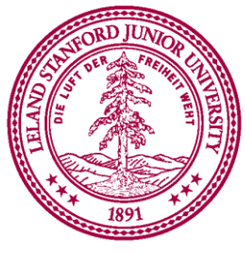
- 300K lines of C++
- Depends on 12 externals packages (MITK, Qt, VTK, ITK, …)
- Build installers for MacOS, Ubuntu and Windows using CMake

Implementation

- No software engineering best practices
- No error handling
- No design documents and no code comments

Software management

- No bug tracking
- No automated testing



Improving SimVascular

Implementation

- Adding implementation documents and code comments
- Replace C-like array data structures with containers
- Remove using object pointers when possible
- Implement error handling using exceptions
- Add unit tests

Software management

- Use GitHub Issues for bug and feature tracking
- Establish strict criteria for merging code
- Jenkins for automated testing
- Add component tests
- Scripting for manual testing of GUI and graphics

Education

Training for for students, post-docs and PI's

- Presentations
- Example code
- Resources

Git and GitHub for sharing and safely storing code

- Workflows: how to use Git in a consistent and productive manner
- Forking and branch strategies to isolate development
- GitHub Issues for providing a history of implementation decisions

Software development

- Software best practices (comments, code structure, etc.)
- Designing for extensibility and maintainability
- Setting lab software standards
- C++ and Python best practices

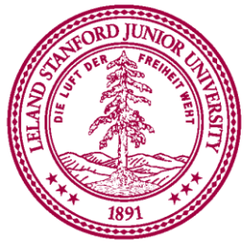
SimVascular Python API

Access SimVascular functionality using Python

- Add functionality not supported by GUI
- Prototyping new methods
- Automate parameter-based generation of models
- Testing core components (modeling, meshing, etc.)

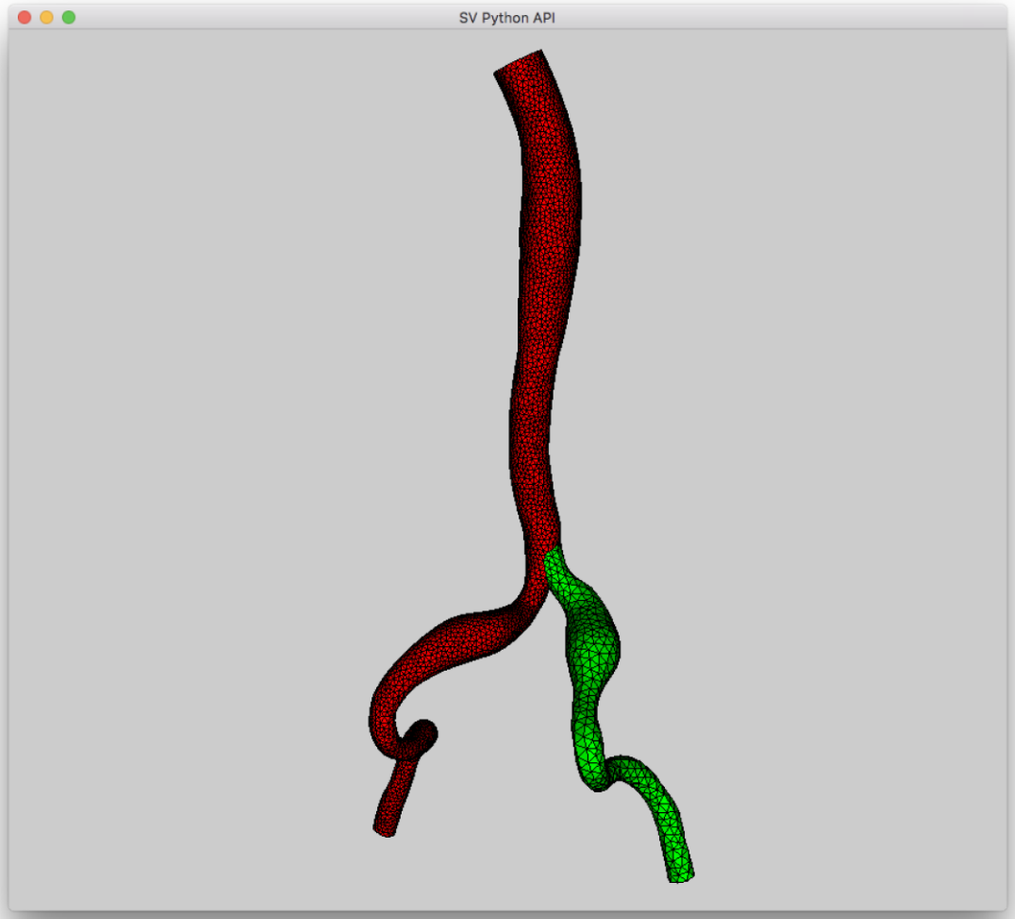
Implementation

- Implemented in C++ using Python C API
- Define Python classes for all SimVascular tools: path planning, segmentation, modeling, meshing and simulation
- Execute scripts in GUI Python console or from the command line
- Visualize geometry using VTK



SimVascular Python API

```
'''Test TetGen local edge (face) refinement.
'''
from pathlib import Path
import sv
import vtk

# Create a TetGen mesher.
mesher = sv.meshing.TetGen()

# Load solid model into the mesher.
home = str(Path.home())
file_name = home + "/SimVascular/DemoProject/Models/demo.vtp"
mesher.load_model(file_name)

# Set the face IDs for model walls.
mesher.set_walls([1, 2])

# Set meshing options.
options = sv.meshing.TetGenOptions(global_edge_size=0.8,  surface_mesh_flag=True,
volume_mesh_flag=True)

# Generate the mesh.
mesher.generate_mesh(options)

# Get the mesh as a vtkUnstructuredGrid.
mesh = mesher.get_mesh()

# Get the mesh surface as VTK PolyData.
mesh_surface = mesher.get_surface()
```



Challenges

Convincing people that good software is important

- Poor design and implementation
  - Produces fragile code that is not usable
  - Wasted time spent trying to understand and modify code
  - Short project life time
- Good software practices can make a big difference

Accountability

- How to enforce software policies
- Post-docs are a big problem

Leadership

- PI’s must take an active role in managing software
- NSF needs to better monitor software coming out of grants

Acknowledgements

Cardiovascular Biomechanics Computation Lab

Funding

National Institutes of Health

NSF

Stanford Children's Health

Computing Resources

XSEDE
Extreme Science and Engineering Discovery Environment