Graduate Student Software Project

[Overview](#_heading=h.30j0zll)

[Project Examples](#_heading=h.1fob9te)

[Model Construction Tool](#_heading=h.3znysh7)

[Model Verification Tool](#_heading=h.2et92p0)

[Model Visualization Tool](#_heading=h.tyjcwt)

[Project Workflow](#_heading=h.1t3h5sf)

[Step 1: Define your tool](#_heading=h.4d34og8)

[Step 2: Determine what existing packages you will use to implement your project.](#_heading=h.2s8eyo1)

[Step 3: Write the Functional Specification](#_heading=h.17dp8vu)

[Step 4: Write the Component Specification](#_heading=h.3rdcrjn)

[Steps 5 and beyond: Iteratively Develop And Refine the Project](#_heading=h.26in1rg)

[Project Structure](#_heading=h.lnxbz9)

[Structure of the Design and Component Specification Documents](#_heading=h.35nkun2)

[Functional Specification](#_heading=h.1ksv4uv)

[Component Specification](#_heading=h.44sinio)

[Project Presentation](#_heading=h.2jxsxqh)

[Grading Rubric](#_heading=h.z337ya)

[Helpful Hints](#_heading=h.3j2qqm3)

[Examples](#_heading=h.1y810tw)

[Useful Links](#_heading=h.4i7ojhp)

# Overview

The project is about developing a python based tool that aids in model construction, verification, and/or visualization. The project should abide by the [Rules for Writing Software](https://docs.google.com/document/d/16ODqwI4uPduJmquQMCBFsmvvzyyRzcZX9GD1d93T9FY/edit#) and should be a python package installable using **just** [**pip**](https://packaging.python.org/tutorials/packaging-projects/). That is, ALL SOFTWARE DEPENDENCIES SHOULD BE HANDLED BY PIP.

# Project Examples

Below are examples of the kinds of tools that you can develop for your software project.

## Model Construction Tool

A model construction tool simplifies the development of a biomedical model.

One example of a tool that aids in model construction is a model preprocessor. This is a program that takes as input a model file with non-model *markups* and outputs a valid model file (e.g., in antimony or SBML). The markups are shorthands for constructs in the model language.

This idea is illustrated below. Consider a preprocessor that simplifies the reaction of kinetics expressions. We consider one example -- simplifying the creation of mass action kinetics in Antimony. Consider a model file that contains the text

A -> B; MA(0.1)

Here MA(0.1) is a markup. This markup indicates that the kinetics law is mass action with a kinetics constant that has the value 0.1. The preprocessor would create a new file that replaces MA(0.1) with k1\*A, and it would add the statement k1 = 0.1. Note that unique constant names must be created for this to work correctly.

## Model Verification Tool

A model verification tool analyzes the reaction network to detect likely errors. Some common errors are mass balance errors, blocked reactions (those that always have zero flux), and orphan chemical species (those that never change from their initial value). Below is an example of a model in which J2 is a blocked reaction and S2, S3 iare an orphan species.

J1: $X -> S1; k1

J2: S2 -> S3; k2\*S2

k1 = 1

k2 = 1

X = 1

S1 = 0

S2 = 0

S3 = 0

Note that you can detect blocked reactions and orphan species by analyzing the stoichiometry matrix.ion

## Model Visualization Tool

Model visualization can relate to the model itself (e.g., a graph of the reaction network) or to the analysis of model results. For example, you might create a tool that aids in parameter estimation by displaying plots that identify which parameters have excessive variabilities or where there might be issues with identifiability of model parameters.

# 

# Project Workflow

## Step 1: Define your tool

Write a short description of the tool that you will build, especially how a user of your tool will interact with the tool. For example, interaction with a model preprocessor tool would be at the command line. The user invokes the preprocessor on a model file with markups and the tool produces a valid model file. Other tools may require programmatic interactions. For example, a tool that visualizes the results of parameter estimation might consist of functions that input observational data and estimated parameter values to produce a series of plots.

In both cases, you will need to package and distribute the tool through the python package installer ([PyPI](https://realpython.com/pypi-publish-python-package/)) so that your package can be installed using pip. For a command line tool, this means that the package includes a command line script. For programmatic interaction tools, this means that you create an importable package.

## Step 2: Determine what existing packages you will use to implement your project.

Almost all modern software is built on existing software packages for visualization, computation, and data access. You might choose a particular project because there are packages that greatly assist your implementation.

## Step 3: Write the Functional Specification

The functional specification details:

\* who are the users and what do they know (e.g., business analyst)

\* what information users want from the system (e.g., where to put bicycles)

\* use cases - how users interact with the system to get the information they want

## Step 4: Write the Component Specification

The component specification describes the components of your software and how they interact to accomplish your use cases. You should include a specification of the data that are input to and output from each major component, and specify how these interactions accomplish at least one use case. An example of a data specification is: a pandas DataFrame with:

1. Columns ``ATP concentration``, ``Glucose concentration``

2. Index: time in seconds

3. Units of values: mM

The sections in the component specification are:

1. Description of components
2. Interactions between components to achieve use cases
3. Preliminary project plan. A plan has time and milestones. Your last milestone is the delivery of your completed project. Provide a detailed milestone for the next 2 weeks and rough milestones that allow you to complete the project on time.

## Steps 5 and beyond: Iteratively Develop And Refine the Project

You will organize the project as a set of short-term deliverables. Typically, you focus first on those parts where you have the most uncertainty since projects typically fail because of "unknown unknowns". For some projects, this will be a data-first focus to make sure that your data can answer the questions that you pose. For others, it may be exploring a python package that you hope will provide key features (e.g., visualizations).

# Project Structure

It is preferred but not required that projects have a [github repository](https://github.com/). The top level directory structure of your project should be organized as follows:

1. README.md file that gives an overview of the project
2. setup.py file that initializes the project after it has been cloned
3. doc folder that contains documentation (including the functional specification, the design specification, and the final project presentation or poster)
4. python package folder (with the same name as the repository) that is structured as one or more python modules (e.g., with \_\_init\_\_.py files) and test files that begin with "test\_".
5. examples folder that contains examples of using the packages

# Structure of the Design and Component Specification Documents

You will create two documents describing the design of your project. These documents should be in your project ``docs`` folder **in a web viewable format (PDF or markdown).**

## Functional Specification

The document should have the following sections:

1. **Background**. The problem being addressed.
2. **User profile**. Who uses the system. what they know about the domain and computing (e.g., can browse the web, can program in Python)
3. **Use cases**. Describe at least two use cases. For each, describe: (a) the objective of the user interaction (e.g., withdraw money from an ATM); and (b) the expected interactions between the user and your system.

## Component Specification

This document should have sections for.

1. **Software components**. High level description of the software components such as: \*data manager\*, which provides a simplified interface to your data and provides application specific features (e.g., querying data subsets); and \*visualization manager\*, which displays data frames as a plot. Describe at least 3 components specifying: what it does, inputs it requires, and outputs it provides.
2. **Interactions to accomplish use cases**. Describe how the above software components interact to accomplish at least one of your use cases.
3. **Preliminary plan**. A list of tasks in priority order.

# Project Presentation

You will present your projects using slides, and a brief demo. The presentation should be about 15 minutes. The presentation should include:

1. Background. Describe the problem or area being addressed.
2. Use cases. How users will interact with your system in a way that addresses the problem area.
3. Demo. Demonstrate your software.
4. Design. Describe the components and how they interact to accomplish the use cases.
5. Project Structure. Show the structure of your github repository.
6. Lessons learned and future work. Focus on \*software engineering\* lessons.

You should post a PDF of your presentation in the docs folder of your project.

# Grading Rubric

You should add Dr. Hellerstein to your GitHub repository. This is done as follows:

1. Browse to your GitHub project
2. Select "Settings" from the GitHub menu
3. Select "Collaborators and teams" from the settings menu
4. Press the green "Add people" button
5. Enter "joseph-hellerstein"

Projects will be evaluated based on the following criteria:

* Project is organized as described in the section on project structure. You can provide a github link or take screenshots of your directory structures.
* Quality of the documentation (especially the functional
* specification and design specification)
* Code quality, especially consistent coding standard (e.g., ``pylint``). Code quality is applied to ALL code in your project, including code copied from others.
* Test coverage. You have written unit tests that execute a large fraction of your code.
* Your package can be installed using PyPI
* Creativity and technical challenge

# Helpful Hints

1. **START EARLY!** These projects take time. Don’t wait.
2. Use virtual environments. A virtual environment allows you to install python packages without conflicting with what else is installed on your machine. This can save you from destroying your machine, and it provides a way for you to test the package you create for your project.
   1. [venv package for python virtual environments](https://docs.python.org/3/library/venv.html)
   2. [Setup a virtual environment on windows 10 (youtube)](https://www.youtube.com/watch?v=vG4AHYXsOgc)
   3. [Setup virtual environment on Mac & linux](https://www.youtube.com/watch?v=iw7KRwRcCwA)

# Examples

The following examples illustrate the software engineering aspects of the project, such as: test files, functional specification, component specification (sometimes referred to as design specification), project directory structure and packaging. Note that the projects per se are unrelated to computational biology.

* [DocstringExpander](https://github.com/joseph-hellerstein/docstring_expander). This is a very small project that illustrates many of the elements of what you’ll need for your course project. The project does not include the functional and component specifications.
* [UWHousingTeam](https://github.com/sliwhu/UWHousingTeam). This project includes functional and component specifications in the docs folder. This is a fairly extensive project that includes much more than what is expected for your course project.
* [PCR Optimizer](https://github.com/Ara101/pcr_optimizer/tree/Version--0.12) A package to help optimize PCR protocols.
* [Antimony Model Creator.](https://github.com/MatthewVanG/AMC-OWMVG) A package that visualizes reactions to aid in creating Antimony models.

# Useful Links

* [Publishing an open source python package](https://realpython.com/pypi-publish-python-package/)
* [Unit tests in python](https://www.datacamp.com/community/tutorials/unit-testing-python)
* [Python documentation and naming conventions](https://realpython.com/python-pep8/)

# Project List

Choose a project from the following list.

1. Design of experiment executor. The user specifies an SBML model, the factors (which should be roadrunner variables such as parameters and initial concentrations), factor levels, and responses (functions of the simulation results). The system does one or more of the following:
   1. Estimates the time to do the study (run all of the experiments)
   2. Performs the experiments
   3. Calculates results for a one-way or two-way design.
2. Parameter estimator. Estimate a subset of parameters in a model. The system should do one or more of the following:
   1. Allow user to describe the parameter estimation problem: which parameters, what range of values.
   2. Run parameter estimation.
   3. VIsualize the results
   4. Update the model with the parameter estimates
3. Network simplify. This project tries to find a simpler model that produces results comparable to those of the original model. Model complexity is quantified by the number of reactions. Two models are equivalent if they produce the same time course. Two models are approximately equal if they have a small RMSE for their time courses.