Standard operating procedure (SOP)   
for multi-scale models

The general workflow should support reproducibility of the results, i.e. with the codecase and the SBML networks it should be possible to fully reproduce the results.

Important to work with the latest codebase, i.e. always pull the latest multiscale-galactose and work with current installations of the tools, i.e. build Copasi & Roadrunner from source regularly.

# 

# SBML model

The first step of the workflow is SBML model generation with subsequent testing of the core model for validity and problems.

## Create models (SBML)

Models are created based on

modelcreator

code from templates.

The code generates the single cell models and the tissue scale model for analysis.

### Annotation?

Here the templates for the sinusoid architecture and the single cell models are combined.

python/modelcreator/models/galactose/GalactoseFactory.py

* TODO: combination of template models with various cell & PKPD models

## Validation & quality control

### SBML validation

All models have to pass the full SBML validation. This is an automatic test step at the end of the file generation

* Check SBML validation results
* no warnings & errors
* full unit check

### Visual inspection & Layout

The SBML files have to be visually inspected based

TODO: automate via cyREST and cy3sbml, i.e. Jupyter Notebook for the visualization

* Load in cy3sbml
* check the annotations, check the provided information and network structure

### Database report

Quality check via inspection of SBML (web interface

manage.py runserver 8001

* are all changes in the SBML represented correctly
* are the event definitions correct
* are the compartments & species, parameters, reactions like expected

### Test simulation

Quality check of SBML via Copasi test simulations (Copasi GUI). Quality and sanity checks if the model can be simulated in an alternative simulation environment with similar results.

* can the model be imported in Copasi?
* are test simulations working for the model? Change some parameters.
* are the results like expected, i.e. events occurring with correct timing, triggers and assignments , no negative concentrations, …)

1. **Create simulation samples**

Define the samples of simulations based on underlying distributions of structural parameters

pyth

* 1. link the Simulations to the correct version of the SBML
  2. create the simulations
  3. check the simulation creation via inspections (web interface)
     1. correct model linked
     2. correct number of simulations
  4. check the parameter file, i.e. export the parameter file for the task and visualize the parameter distributions underlying the SimulationTask. Is this what should be simulated?

### Run simulations on Clients

* 1. commit latest source ! to guarantee identical code base on clients
  2. distribute SBML to clients (**syncDjangoSBML.sh** & **syncSBML.sh**)
  3. ./startSimulation (Clients start listening to available simulations and run them)

### 6. Check individual integration

Check single integration via visualization, especially the boundary conditions (open CSV and see what happend. Is this what should be simulated - focus on the defined events).

* some analysis functions for single simulation are available (SingleSimulation.plot?)

### Collect data for post-processesing

The simulation definitions and integration results have to integrated.

* All data from the multiple sample for one simulation have to be brought together. Collection of the necessary data from the clients (**PrepareAnalysisBatch.py** & **prepareDataForAnalysis.py**)
* Convert data to R structures - Rdata files are directly generated after the simulation to save postprocessing time

### Handling Django static files

TODO A problem is the collection of the static files. Depending on which computer is running the files are stored locally.

This could be solved via creating sim links to folders.

Main problem are the sbml files.

scp dirty solution

scp -r ~/multiscale-galactose-results/django/sbml mkoenig@10.39.32.111:~/multiscale-galactose-results/django/

TODO create a startscript which performs the necessary steps

-> checkout latest source

-> recompile the CopasiModelRunner make

-> create the temporary folders for the simulation

i.e. ~/multiscale-galactose-results/tmp\_sim

TODO create makefile /cmake file for the project.

# Simulation analysis

## Galactose Challenge

### GEC curves

Create the GEC curves via combining the simulations for given galactose challenges and perfusion scaling. Implemented

Use **GEC\_curves.R** to calculate the GEC curves with bootstrap results.

Analysis of GEC based on simulations

* make some visual inspections of the resulting blood flows and galactose clearance (is this correct? expected distributions?)
* add additional values calculated based on SBML information (namely volumes, areas, … Parameters which are defined in the SBML and are used in combination with the timecourse data to calculate derived values

### GEC predictions

Use the calculated GEC curves for prediction.

**GEC\_prediction.R** - Here liver volumes, blood flows and GEC are predicted.

On the one hand for the experimental data, on the other hand for the

**GEC\_NHANES\_analysis.R**

## Multiple-Indicator Dilution curves

# Iteration cycle GEC

Cycle 1 -> Version 25 [2014-11-15]

* prediction of GEC is too high 20% in general after accounting for the vessel geometry
* rescaling of the f\_scale for the metabolism from 6.6E-15 to 5.3E-15