# MultiExperiments Class

Fields

Modele* m;
int NbBigExps;
vector<Experiment*> ListBigExperiments;

Input/Output methods

MultiExperiments(Modele* _m);
nbBigExp();
void AddExperiment(Experiment* Ex);
Experiment* getExperiment(int BigExpID);
string BigExpName(int i);

void init();
virtual void simulate(int IdExp, Evaluator* E = NULL, bool force = false);
void simulateAll(bool force = false);
virtual double costPart(int speciesGlobalID = -1);
virtual double costBigExp(int BigExpID = -1);
void reset();
string print();
double getPenalities();
void overrideVariable(int IdGlobalName, bool override);
string costReport();

# Experiment Class

Fields

Modele* m;
string Identification; -> name
int nbConditions; -> nb of sub-experiments
vector<string> names_exp;
vector<bool> doable;
-> depending on the model, list of doable experiments
vector<Evaluator*> VTG;
-> list of Values To Get from a simulation
vector<overrider*> Overs;
double totalPenalities;

vector<TableCourse*> ExpData;
vector<TableCourse*> ExpStdDevs;
vector<string> NamesVariablesInTheirIndex;

Input/Output methods

Experiment(Modele* _m, int _nbConditions);
int nbCond();
string expName(int i);
bool isDoable(int i);
double V(int IDexp, int Species, int time_sec)
-> calls VTG[IDexp]->getExpVal(m->internValName(Species), time_sec)
void setOverrider(int IdExp, overrider* _ov = NULL);
void overrideVariable(int IdGlobalName, bool override = true, int IdExp = -1);
bool canOverride(int IdGlobalVariable, int IdExp = -1);
-> say if the overriders have data for a variable
double getPenalities();

void giveData(TableCourse* kineticData, int IdExp);
void giveStdDevs(TableCourse* kineticStds, int IdExp);
void giveHowToReadNamesInKinetics(vector<string> _NamesVariablesInTheirIndex);
void loadEvaluators();

functions that a (sub-) experiment SHOULD re-implement

virtual void simulate(int IdExp, Evaluator* E = NULL, bool force = false);

pre-implemented functions :

virtual void init();
-> clears all the evaluators VTG[] but keeping experimental data
bool motherPrepareSimulate(int IdExp, Evaluator *&E, bool force);
-> set the evaluator in acquisition mode (E->recordingCompleted())
-> sets the overriders into the model, for this experiment
by calling m->setOverrider(Overs[IdExp]);
virtual void simulateAll(bool force = false)
-> for all condition i, calls simulate(i, VTG[i], force)
virtual double costExp(int ExpID = -1)
virtual double costPart(int speciesGlobalID = -1)
-> both these functions sum the costs between simulation and data (by exp. or by variable)
by calling VTG[expID]->getFitnessSpecies(internalValName(speciesGlobalID)));
-> note : the simulation penalities are not included in this cost.
virtual void reset()
-> clears everything

string print();
string extractData(vector<int> timePoints, vector<int> idGlobalVariables, vector<string> namesAllGlobVars, vector<int> IDexps = vector<int>());