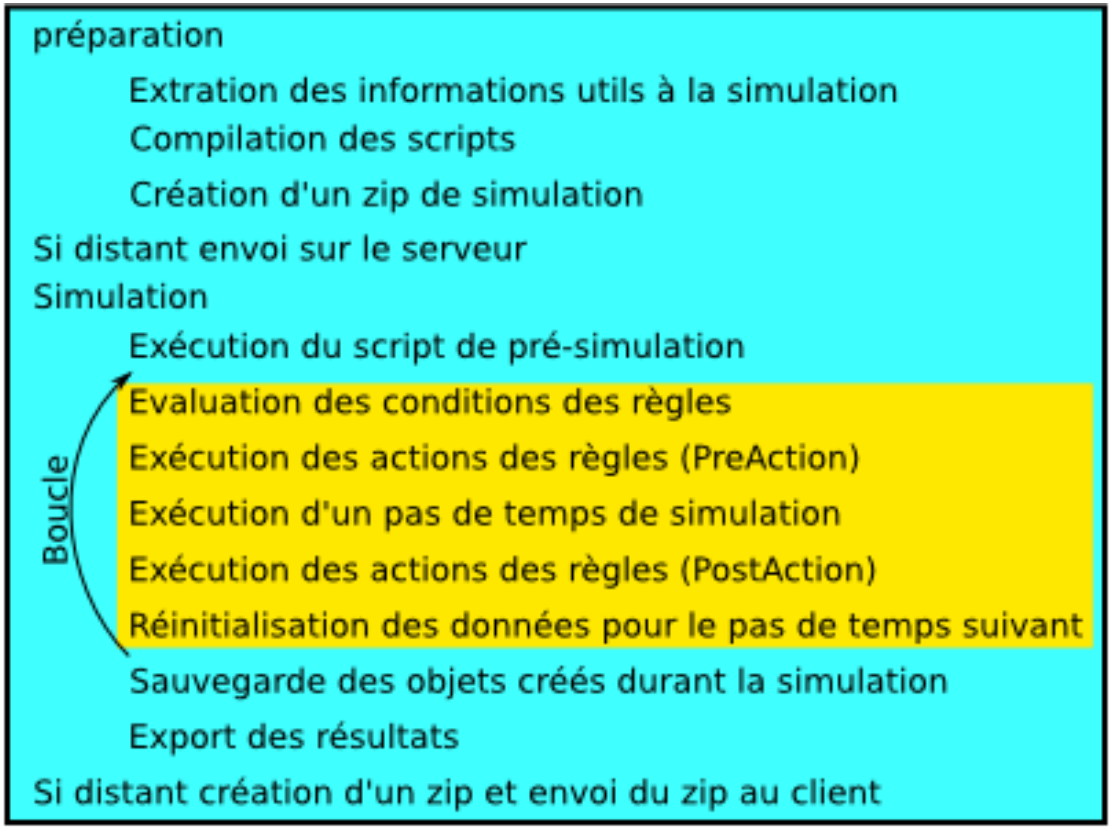
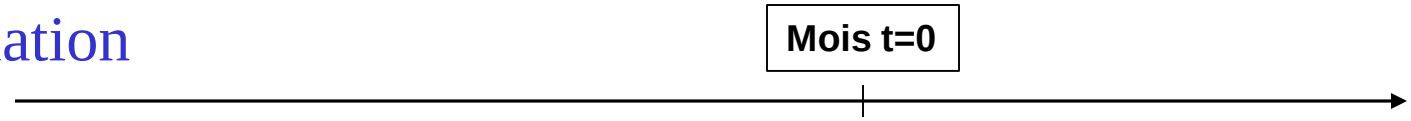
Default Simulator
Rules
Train of processes (chronology)

A simulation with ISIS-Fish



DefaultSimulator

presimulation



SimulationContext
SimulationStorage()
SimulationControl()
Step()
DB()
PopulationMonitor()
MetierMonitor()
RuleMonitor()

Default simulator

```
public class DefaultSimulator implements Simulator {

   static private Log log = LogFactory.getLog(DefaultSimulator.class);

   public void simulate(SimulationContext context) throws Exception {}

  protected boolean isEffortByCell(SimulationContext context) {}

  protected void computeMonth(SimulationContext context, SiMatrix
siMatrix,TimeStep step, Population pop) throws IsisFishException,
TopiaException {}

  private void saveGravityModel(TimeStep step, ResultManager
resManager, GravityModel gravityModel) throws IsisFishException,
TopiaException {}
}
```

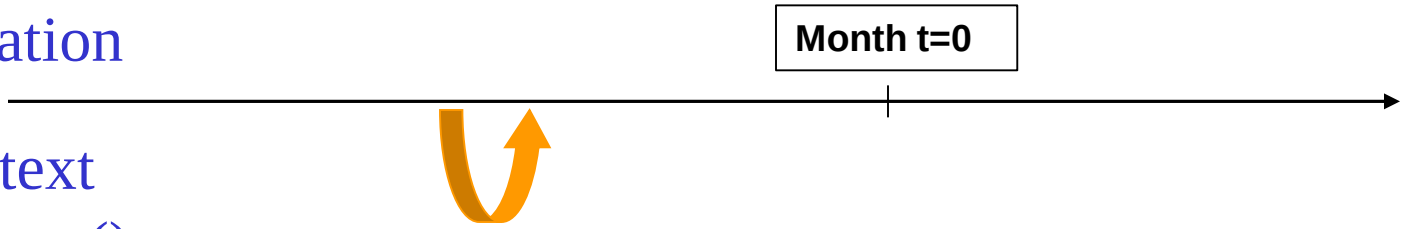
Default simulator

```
public class DefaultSimulator implements Simulator {

   static private Log log = LogFactory.getLog(DefaultSimulator.class);

   public void simulate(SimulationContext context) throws Exception {}

   protected boolean isEffortByCell(SimulationContext context) {}

   protected void computeMonth(SimulationContext context, SiMatrix
      siMatrix, TimeStep step, Population pop) throws IsisFishException,
                                                      TopiaException {}

   private void saveGravityModel(TimeStep step, ResultManager
            resManager, GravityModel gravityModel) throws
                                          IsisFishException, TopiaException {}
}
```

simulate(SimulationContext context)

```
SimulationParameter param = context.getSimulationStorage().getParameter();
SimulationControl control = context.getSimulationControl();
int lastYear = param.getNumberOfYear(); int lastStep = lastYear month.NUMBER_OF_MONTH;
TimeStep step = control.getStep();
ResultManager resManager = context.getResultManager();
TopiaContext db = context.getDB();
SiMatrix siMatrix = SiMatrix.getSiMatrix(context);
GravityModel gravityModel = new GravityModel(context, siMatrix);
PopulationMonitor populationMonitor = context.getPopulationMonitor();
MetierMonitor metierMonitor = context.getMetierMonitor();
RuleMonitor ruleMonitor = context.getRuleMonitor();
List<Population> allpops = siMatrix.getPopulations(step);
populationMonitor.init(allpops);

for (Population pop : allpops) {
MatrixND N = param.getNumberOf(pop);  N.setName(ResultName.MATRIX_ABUNDANCE);
populationMonitor.setN(pop, N);       }

param.reloadContextParameters();

// Rule initialisation       //
 List<Rule> rules = param.getRules(); control.setText("Rules initialisation:" + rules);
 for (Rule rule : rules) { rule.init(context);}

// Commit all change done un init rules methods.       //
context.getDB().commitTransaction();
```

presimulation



SimulationContext
SimulationStorage()
SimulationControl()
Step()
DB()
PopulationMonitor()
MetierMonitor()
RuleMonitor()

Initializing rule
Loop rule

simulate(SimulationContext context)

```
// Simulation loop        //
 while (step.getStep() < lastStep) {
```

```
  rules = param.getRules();
  metierMonitor.clear();   // annulation de l'effet des règles du pas de temps précédent (Valeurs de BD)
  if (step.getMonth().equals(Month.JANUARY)) {  populationMonitor.clearCatch(); }

// Rule condition evaluation          //
for (Rule rule : rules) {
 for (Metier metier : siMatrix.getMetiers(step)) {
    boolean active = false;
     try {active = rule.condition(context, step, metier);}
    catch (Exception eee) {if (log.isWarnEnabled()) {log.warn("Can't evaluate rule condition for: " + rule, eee);
   ruleMonitor.setEvaluationCondition(step, rule, metier? active);
   if (active) {resManager.addActiveRule(step, rule); }
 }
}

// Rule pre action          //
for (Rule rule : rules) {
for (Metier metier : siMatrix.getMetiers(step)) {
 boolean condition = ruleMonitor.getEvalutionCondition(step, rule, metier);
  if (condition) {rule.preAction(context, step, metier);  }
 }
}
```

1. Assess rule conditions

Metiers loop

2. Changes before rule application

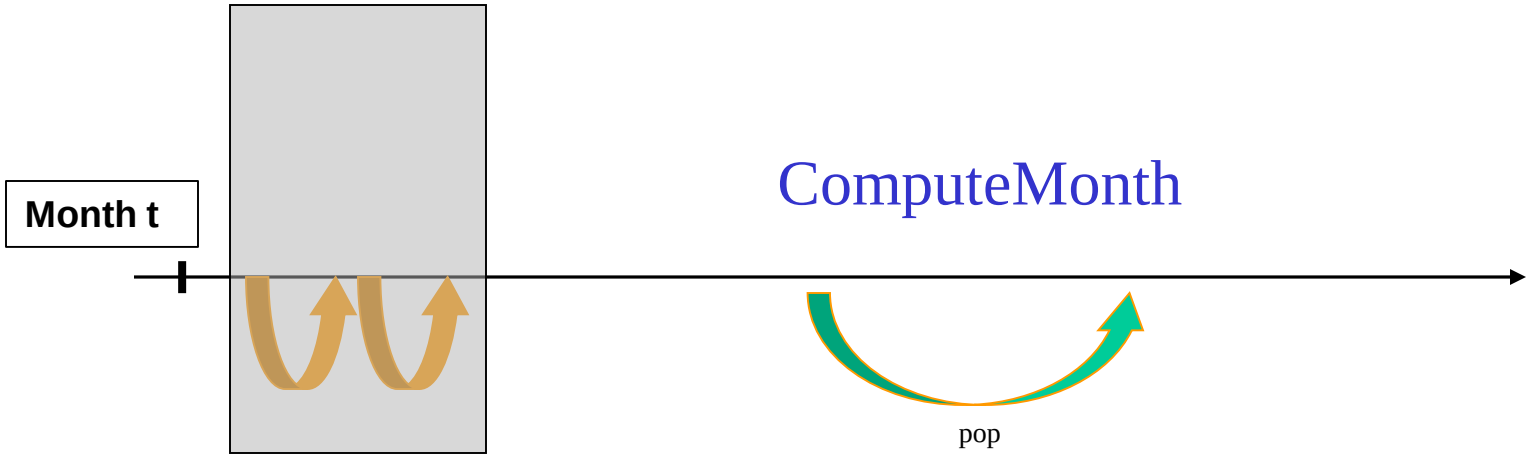
Metiers (condition= TRUE) loop

Changement de zones metier

Changement PropStr….

simulate(SimulationContext context)

```
// Simulation loop        //
 while (step.getStep() < lastStep) {

// Keep modification's information done in rule          //
 if (resManager.isEnabled(ResultName.MATRIX_METIER_ZONE)) {
    MatrixND metierZone = siMatrix.getMetierZone(step); resManager.addResult(step, metierZone);          }

// Simulate one step for all pop          //
control.setText("Simulate one month");
 for (Population pop : siMatrix.getPopulations(step))
            {computeMonth(context, siMatrix, step, pop); }
```


Month t
ComputeMonth
pop

Default simulator

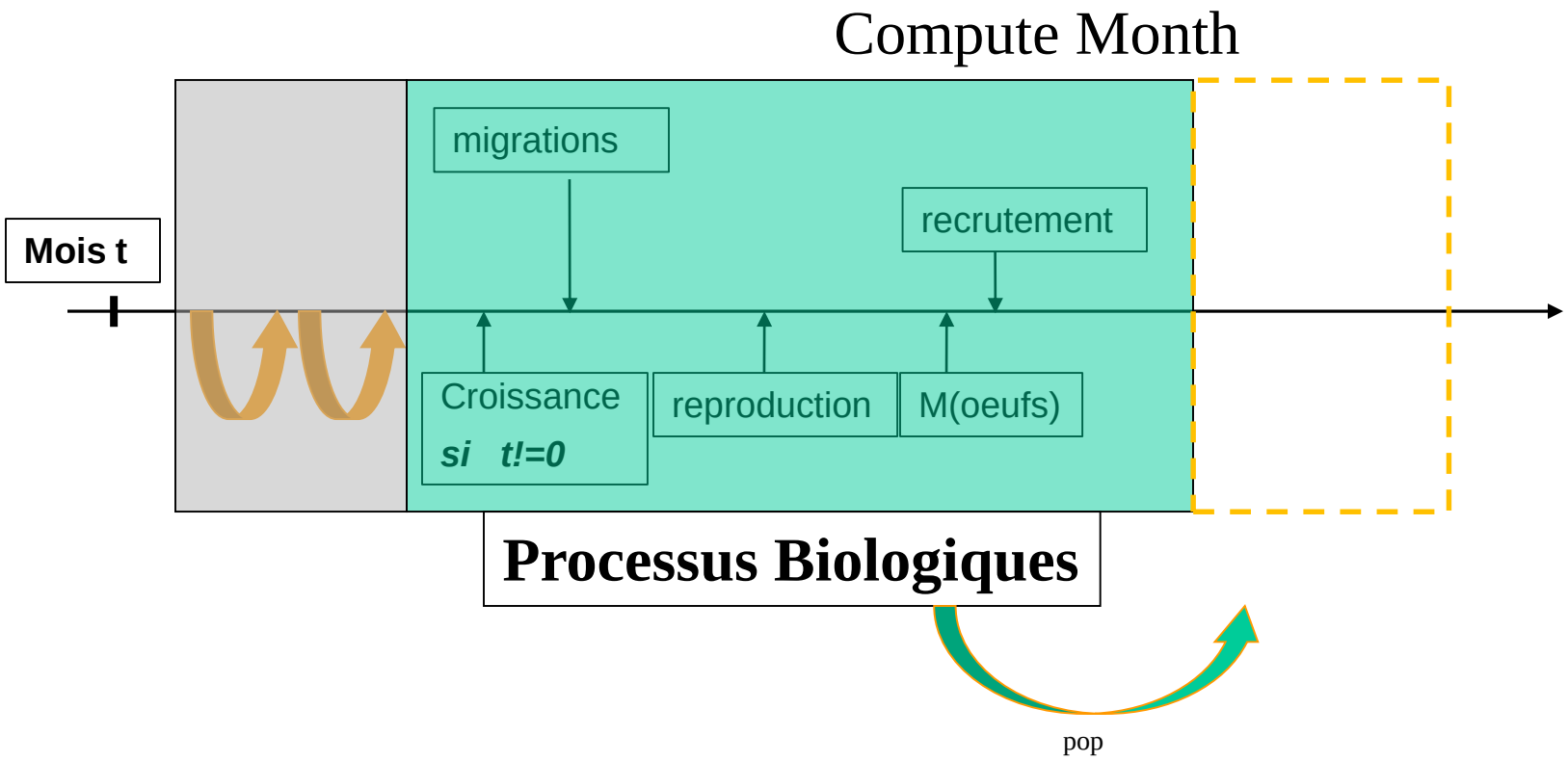
```
public class DefaultSimulator implements Simulator {

  static private Log log = LogFactory.getLog(DefaultSimulator.class);

  public void simulate(SimulationContext context) throws Exception {}

  protected boolean isEffortByCell(SimulationContext context) {}

  protected void computeMonth(SimulationContext context, SiMatrix
siMatrix,TimeStep step, Population pop) throws IsisFishException,
TopiaException {}

  private void saveGravityModel(TimeStep step, ResultManager
resManager, GravityModel gravityModel) throws IsisFishException,
TopiaException {}

}
```

computeMonth(SimulationContext context, SiMatrix siMatrix, TimeStep step, Population pop)

```
// to add result
ResultStorage resManager = context.getSimulationStorage().getResultStorage();
PopulationMonitor popMon = context.getPopulationMonitor();
MatrixND N = popMon.getN(pop);

// add N and biomass result now, before computation
// N is reassigned during computation
resManager.addResult(step, pop, N); // Stockage de N avant Biologie (visualisation interface resultage)
if (resManager.isEnabled(ResultName.MATRIX_BIOMASS)) {// Stockage de B avant Biologie
 MatrixND biomass = siMatrix.matrixBiomass(N, pop, step);
 resManager.addResult(step, pop, biomass);       }

Month month = step.getMonth();
PopulationSeasonInfo info = pop.getPopulationSeasonInfo(month);

// group change
MatrixND CA;
 if (step.getStep() == 0) {
     CA = MatrixFactory.getInstance().matrixId(pop.sizePopulationGroup() * pop.sizePopulationZone());  }
 else {  CA = info.getGroupChangeMatrix(month);        }
```

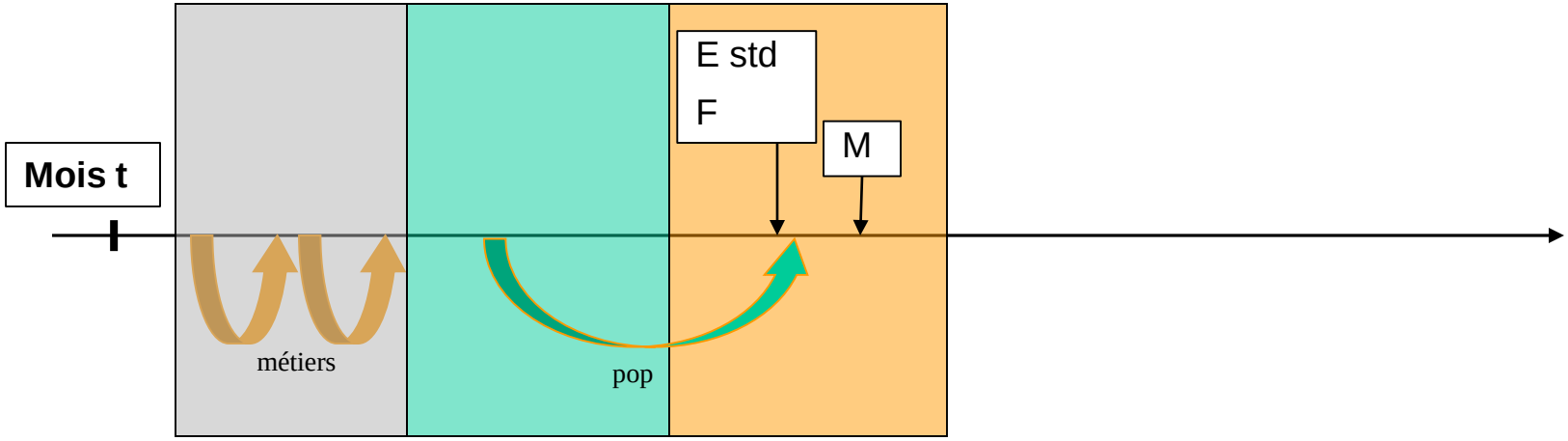
computeMonth(SimulationContext context, SiMatrix siMatrix, TimeStep step, Population pop)

```
//migration
MatrixND M = info.getMigrationMatrix(month, N);
//emigration
MatrixND EM = info.getEmigrationMatrix(month, N);
//immigration
 MatrixND IM = info.getImmigrationMatrix(month, N).transpose();
// First computations with N (abundance) put in matrix 1D
MatrixND N1D = pop.N2DToN1D(N);
MatrixND tmp0 = N1D.mult(CA);
MatrixND tmp1 = M.minus(EM);
MatrixND tmp2 = tmp0.mult(tmp1);
MatrixND tmp3 = tmp2.add(IM);
N = pop.split2D(tmp3); // On reconvertie en une matrice Semantique
// reproduction
MatrixND R = info.getReproductionMatrix(month, N);
//  adding  R matrix in pop object the reproduction of the month
popMon.setReproduction(step, pop, R)
// recruitement
MatrixND recruitment = popMon.getRecruitment(step, pop);
popMon.applyReproductionMortality(pop); // mortality of eggs (larvae)
N = N.add(recruitment);
```


Compute Month
migrations
recrutement
Mois t
Croissance
si t!=0
reproduction
M(oeufs)
Processus Biologiques
pop

computeMonth(SimulationContext context, SiMatrix siMatrix, TimeStep step, Population pop)

```
if (resManager.isEnabled(ResultName.MATRIX_ABUNDANCE_BEGIN_MONTH)) {
    MatrixND abondanceBM = siMatrix.matrixAbondanceBeginMonth(N, pop, step);
    resManager.addResult(step, pop, abondanceBM);       }
if (resManager.isEnabled(ResultName.MATRIX_BIOMASS_BEGIN_MONTH)) {
    MatrixND biomassBM = siMatrix.matrixBiomassBeginMonth(N, pop, step);
    resManager.addResult(step, pop, biomassBM);       }
MatrixND abundance;
// compute fishing matrix only if there are one or more strategy
if (siMatrix.getStrategies(step).size() > 0) {
// compute some Matrix and add result
MatrixND catchPerStrategyMetPerZonePop; // this matrix is necessary for PopulationMonitor.holdCatch
                                                                                    (reused in rule)
 if (isEffortByCell(context)) {
  abundance = siMatrix.matrixAbundance(N, pop, step);
  catchPerStrategyMetPerZonePop = siMatrix .matrixCatchPerStrategyMetPerZonePop(N, pop, step);  }
 else { // en zone
   MatrixND matrixFishingMortality = siMatrix .matrixFishingMortality(step, pop);
   resManager.addResult(step, pop, matrixFishingMortality);
   abundance = siMatrix.matrixAbundance(N, pop, step, matrixFishingMortality);
   MatrixND catchRatePerStrategyMet = siMatrix.matrixCatchRatePerStrategyMetPerZone(pop, step,
                                                                    matrixFishingMortality);
  resManager.addResult(step, pop, catchRatePerStrategyMet);
  catchPerStrategyMetPerZonePop = siMatrix .matrixCatchPerStrategyMetPerZone(N, pop, step,
                                                                        catchRatePerStrategyMet);
 }
```



Computation of

- Abundance
- FishingMortality
- CatchPerStrategyMetPerZonePop
- and associated indicators

computeMonth(SimulationContext context, SiMatrix siMatrix, TimeStep step, Population pop)

```
popMon.holdCatch(pop, catchPerStrategyMetPerZonePop);
resManager.addResult(step, pop, catchPerStrategyMetPerZonePop);
 if (resManager.isEnabled(ResultName.MATRIX_CATCH_WEIGHT_PER_STRATEGY_MET_PER_ZONE
  MatrixND catchWeightPerStrategyMet = siMatrix .matrixCatchWeightPerStrategyMetPerZonePop(step, pop,
                                                                    catchPerStrategyMetPerZonePop);
  resManager.addResult(step, pop, catchWeightPerStrategyMet);              }
if (isEffortByCell(context)) {
  MatrixND catchPerStrategyMetPerZoneMet = siMatrix.matrixCatchPerStrategyMetPerZoneMet(N, pop, step);
 resManager.addResult(step, pop, catchPerStrategyMetPerZoneMet);
 if (resManager.isEnabled(ResultName.MATRIX_CATCH_WEIGHT_PER_STRATEGY_MET_PER_ZONE_MET)) {
     MatrixND catchWeightPerStrategyMet = siMatrix.matrixCatchWeightPerStrategyMetPerZoneMet(step,
                  pop, catchPerStrategyMetPerZoneMet);
    resManager.addResult(step, pop, catchWeightPerStrategyMet);            }
}
if (resManager.isEnabled(ResultName.MATRIX_FISHING_MORTALITY_PER_GROUP) ||
  resManager.isEnabled(ResultName.MATRIX_TOTAL_FISHING_MORTALITY)) {
  MatrixND fishingMortalityPerGroup = siMatrix.fishingMortalityPerGroup(step, pop,
                                    context.getSimulationStorage().getResultStorage());
   if (resManager.isEnabled(ResultName.MATRIX_FISHING_MORTALITY_PER_GROUP)) {
              resManager.addResult(step, pop, fishingMortalityPerGroup);              }
   if (resManager.isEnabled(ResultName.MATRIX_TOTAL_FISHING_MORTALITY)) {
            MatrixND totalFishingMortality = siMatrix.totalFishingMortality(step, pop, fishingMortalityPerGro
           resManager.addResult(step, pop, totalFishingMortality);              }        }
 } else { // no strategies  : compute only if fishing mortality =0 to apply Natural Mortality
 abundance = siMatrix.matrixAbundanceSsF(N, pop, step);       }
popMon.setN(pop, abundance); // Keep new N
```

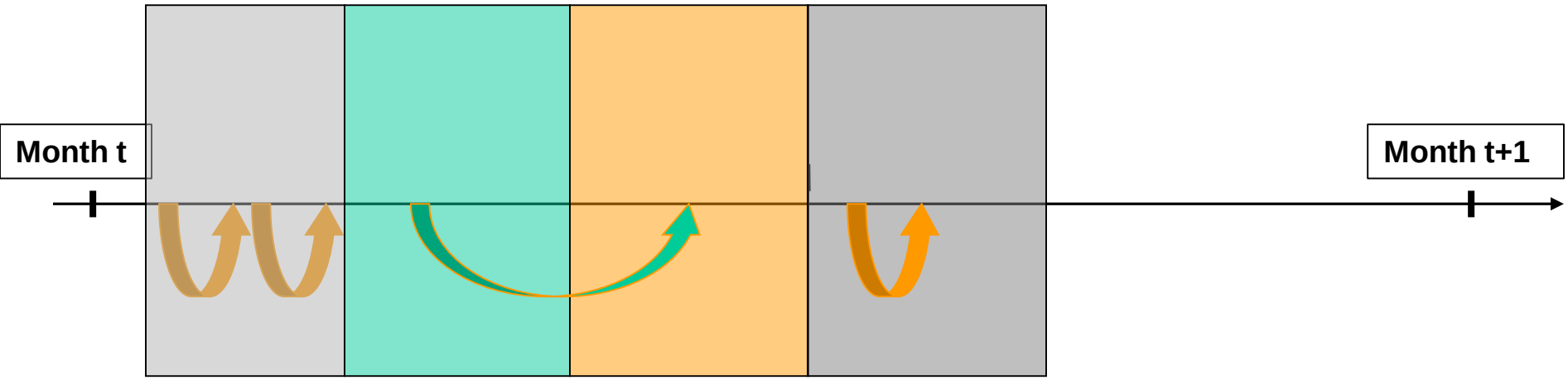
simulate(SimulationContext context)

```
// Simulation loop        //
 while (step.getStep() < lastStep) {
```

```
// Keep modification's information done in rule          //
 if (resManager.isEnabled(ResultName.MATRIX_METIER_ZONE)) {
MatrixND metierZone = siMatrix.getMetierZone(step); resManager.addResult(step, metierZone);          }

// Simulate one step for all pop          //
control.setText("Simulate one month");
 for (Population pop : siMatrix.getPopulations(step))
            {computeMonth(context, siMatrix, step, pop); }

// Add some result not population dependante          //
if (siMatrix.getStrategies(step).size() > 0) {
 if (resManager.isEnabled(ResultName.MATRIX_EFFORT_PER_STRATEGY_MET)) {
       MatrixND effortPerStrategyMet = siMatrix .matrixEffortPerStrategyMet(step);
       resManager.addResult(step, effortPerStrategyMet);              }
if (resManager.isEnabled(ResultName.MATRIX_EFFORT_NOMINAL_PER_STRATEGY_MET)) {
      MatrixND effortNominalPerStrategyMet = siMatrix.matrixEffortNominalPerStrategyMet(step);
      resManager.addResult(step, effortNominalPerStrategyMet);              }
```

simulate(SimulationContext context)

```
// Simulation loop        //
 while (step.getStep() < lastStep) {

// Rule post action           //
for (Rule rule : rules) {
   for (Metier metier : siMatrix.getMetiers(step)) {
 if (ruleMonitor.getEvalutionCondition(step, rule, metier)) {rule.postAction(context, step, metier); } } }

// discard and landing must be done after post action rules
for (Population pop : siMatrix.getPopulations(step)) {
    MatrixND discard = populationMonitor.getDiscard(step, pop);
    if (discard != null || step.getStep() == 0) { // force discard for the first month to have discard in result
     if (discard == null) {
       discard =MatrixFactory.getInstance().create(ResultName.MATRIX_DISCARDS_PER_STR_MET_PER_ZONE_POP,
                        new List[] { siMatrix.getStrategies(step), siMatrix.getMetiers(step), pop.getPopulationGroup(),
                        pop.getPopulationZone() }, new String[] { n("Strategies"), n("Metiers"), n("Groups"), n("Zones") }); }
      resManager.addResult(step, pop, discard);
   if (resManager.isEnabled(ResultName.MATRIX_DISCARDS_WEIGHT_PER_STR_MET_PER_ZONE_POP)) {
  MatrixND discardWeightPerStrategyMet = siMatrix.matrixDiscardWeightPerStrategyMetPerZonePop(pop, step, discard);
   resManager.addResult(step, pop, discardWeightPerStrategyMet);                    }                    }
```



Changes after rule application

Metiers loop

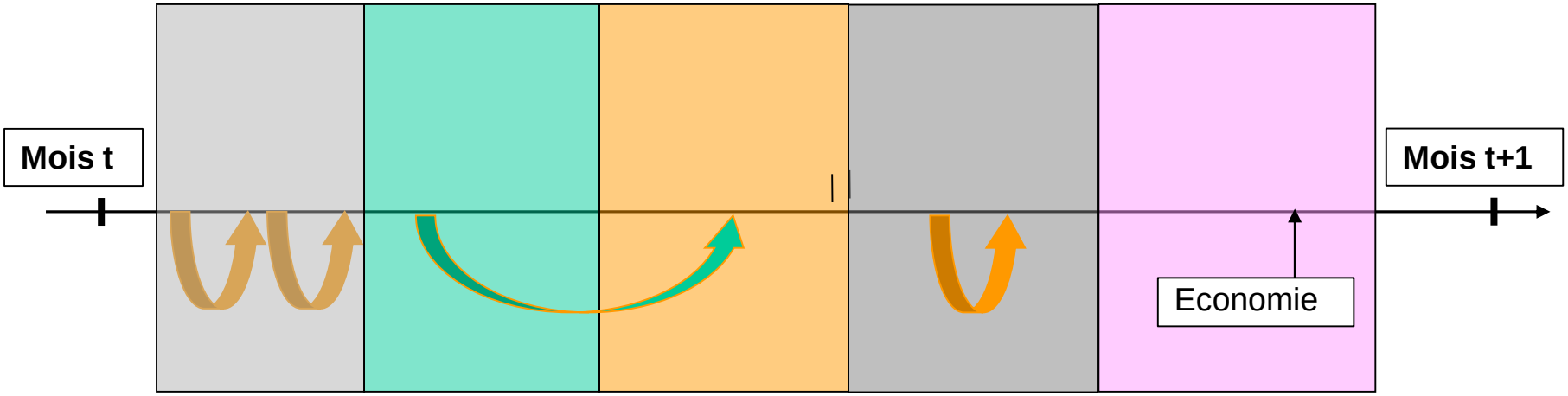
Computation of
- discards
- landings
- abundance

simulate(SimulationContext context)

// Simulation loop //
while (step.getStep() < lastStep) {

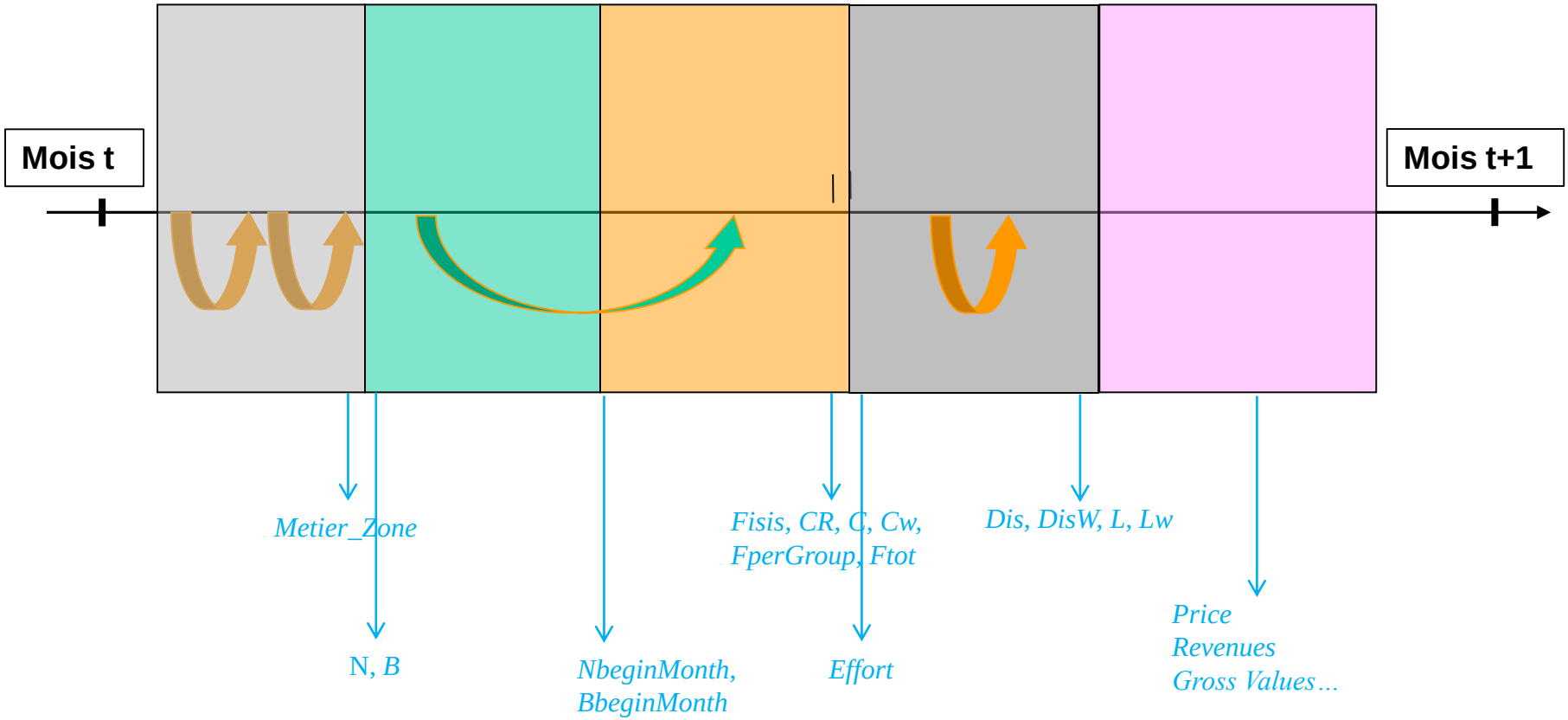
// Add economics results // after post-action because revenus are computed with landings and not catches and price equation can be linked to landings
if (!"false".equalsIgnoreCase(param.getTagValue().get("ecoResult"))) {
saveGravityModel(step, resManager, gravityModel); } // includes ResManager and GravityModel computation

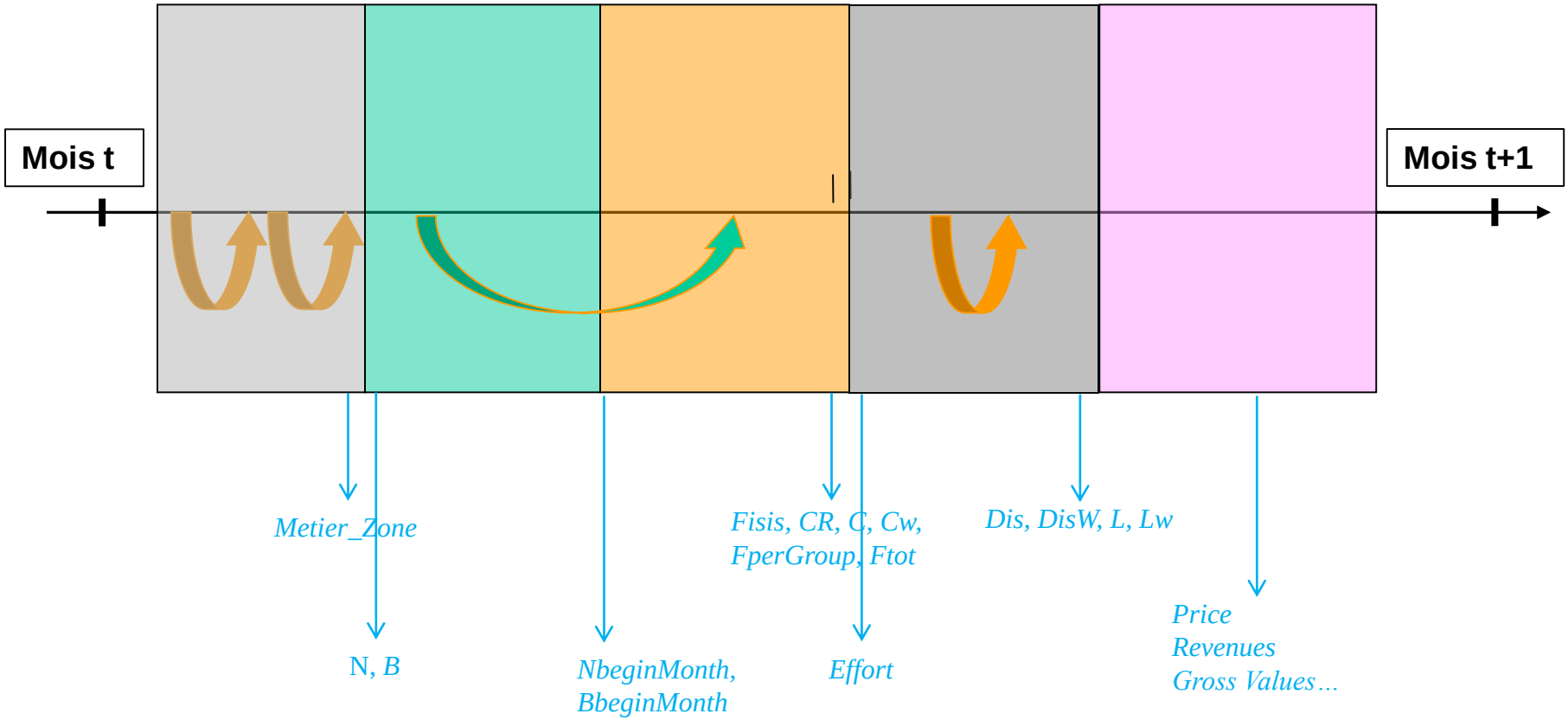
// Add economics results
if (resManager.isEnabled(ResultName.MATRIX_PRICE)) {
for (Population pop : siMatrix.getPopulations(step)) {
MatrixND matPrice = siMatrix.matrixPrice(step, pop);
resManager.addResult(step, pop, matPrice); } }



- Price
- Economics

Computation of economic indicators


Mois t
Mois t+1
Metier_Zone
N, B
NbeginMonth,
BbeginMonth
Fisis, CR, C, Cw,
FperGroup, Ftot
Effort
Dis, DisW, L, Lw
Price
Revenues
Gross Values…


Mois t
Mois t+1
Metier_Zone
N, B
NbeginMonth,
BbeginMonth
Fisis, CR, C, Cw,
FperGroup, Ftot
Effort
Dis, DisW, L, Lw
Price
Revenues
Gross Values…

simulate(SimulationContext context)

```
// Simulation loop        //
 while (step.getStep() < lastStep) {

// revert modification for next step
control.setText("Rollback rules changes");
db.rollbackTransaction();

 // commit result
control.setText("Commit results");
 TopiaContext tx = context.getDbResult();
tx.commitTransaction();

// Go next step
 step = step.next();

}
```