PhysicalNode.java

```
/**
 * Degradate (according to degrad. constant) and diffuse 8according to diff. constant)
 * all the <code>Substance</code> stored in this <code>PhysicalNode</code>.
 */
public void runExtracellularDiffusion(){
        // 1) now that we are about to diffuse, a new diffusion should only be performed
        // if there is a good reason for that.
        onTheSchedulerListForPhysicalNodes = false;
        double currentEcmTime = ecm.getECMtime();
        // 2) Degradation according to the degradation constant for each chemical
        degradate(currentEcmTime);
        // 3) Diffusion (along every edge)
        for (SpatialOrganizationEdge<PhysicalNode> e : soNode.getEdges()) {
          diffuseEdgeAnalytically(e, currentEcmTime);
        }
}

/**
 * Runs the degradation of all Substances (only if it is not up-to-date). This method
 * is called before each operation on Substances (
 * @param currentEcmTime the current time of the caller
 * (so that it doesn't require a call to ECM).
 */
protected void degradate(double currentEcmTime){ //changed to proteceted

        // if we are up-to-date : we stop here.
        if(lastECMTimeDegradateWasRun > currentEcmTime){
          return;
        }

        //getRwLock().writeLock().lock();
        // Otherwise, degradation according to the degradation constant for each chemical
        double deltaT = currentEcmTime-lastECMTimeDegradateWasRun;
        for (Substance s : extracellularSubstances.values()) {
          double decay = Math.exp(-s.getDegradationConstant()*deltaT);
          s.multiplyQuantityAndConcentrationBy(decay);
        }

        // We store the current time as the last time we updated degradation
        // (+0.0000001, to be on the safe side of the double comparison)
        lastECMTimeDegradateWasRun= currentEcmTime+0.0000001;
        //
}

/* Analytic solution of the diffusion process along the edge between two PhysicalNodes.
* dQA/dt = diffCst*(Area/distance)*(QB/VB-QA/VA)
```

```
 */
private void diffuseEdgeAnalytically(SpatialOrganizationEdge<PhysicalNode> e, double
currentEcmTime) {

        // the two PhysicalNodes
        PhysicalNode nA = this;
        PhysicalNode nB = e.getOppositeElement(this);



        // make sure the other one is up-to-date with degradation
        nB.degradate(currentEcmTime);
        // some values about space node distances, contact area and volume
        SpatialOrganizationNode<PhysicalNode> sonA = getSoNode();
        SpatialOrganizationNode<PhysicalNode> sonB = nB.getSoNode();
        double distance = distance(sonA.getPosition(), sonB.getPosition());
        double vA = sonA.getVolume();
        double vB = sonB.getVolume();
        double pre_a = (e.getCrossSection()/distance);
        double pre_m = (e.getCrossSection()/distance) * (1.0/vA + 1.0/vB);

        // diffusion of all the extracellularSubstances in A :
        for (Enumeration<Substance> substancesEnumeration =
      nA.extracellularSubstances.elements(); substancesEnumeration.hasMoreElements();) {
                // for a given substance
                Substance sA = substancesEnumeration.nextElement();
                double sAConcentration = sA.getConcentration();
                // stop here if 1) non diffusible substance or 2) concentration very low:
                double diffusionConstant = sA.getDiffusionConstant();
                if(diffusionConstant<10E-14 ||
sAConcentration<Param.MINIMAL_CONCENTRATION_FOR_EXTRACELLULAR_DIFFUSION){
                        continue; // to avoid a division by zero in the n/m if the diff const = 0;
                }
                // find the counterpart in B
                Substance sB = nB.getSubstanceInstance(sA);
                double sBConcentration = sB.getConcentration();
                // saving time : no diffusion if almost no difference;
                double absDiff = Math.abs(sAConcentration-sBConcentration);

if( (absDiff<Param.MINIMAL_DIFFERENCE_CONCENTRATION_FOR_EXTRACELLULAR_DIFFU
SION) ||
(absDiff/sAConcentration<Param.MINIMAL_DC_OVER_C_FOR_EXTRACELLULAR_DIFFUSION))
{
                        continue;
                }
                // If we reach this point, it means that it is worth performing the diffusion.
                // we thus put ourselves on the list for performing it again next time step.
                nB.setOnTheSchedulerListForPhysicalNodes(true);
                this.onTheSchedulerListForPhysicalNodes = true;
```

```
            // Analytical computation of the diffusion between these two nodes
            // (cf document "Diffusion" by F.Zubler for explanation).

            double qA = sA.getQuantity();
            double qB = sB.getQuantity();
            double Tot = qA + qB;
            double a = pre_a*diffusionConstant;
            double m = pre_m*diffusionConstant;

            double n = a*Tot/vB;
            double nOverM = n/m;
            double K = qA -nOverM;

            qA = K*Math.exp(-m*Param.SIMULATION_TIME_STEP) + nOverM;
            qB = Tot - qA;

            sA.setQuantity(qA);
            sB.setQuantity(qB);
            // and update their concentration again
            sA.updateConcentrationBasedOnQuantity(vA);
            sB.updateConcentrationBasedOnQuantity(vB);
    }

}


/** Modifies the quantity (increases or decreases) of an extra-cellular Substance.
 * If this <code>PhysicalNode</code> already has an <code>Substance</code> instance
 * corresponding to the type given as argument (with the same id), the fields
 * quantity and concentration in it will be modified, based on a computation depending
 * on the simulation time step and the space volume. If there is no such Substance
 * instance already, a new instance is requested from ECM.
 * <p>
 * This method is not used for diffusion, but only by biological classes...
 * @param id the name of the Substance to change.
 * @param quantityPerTime the rate of quantity production
 */
public void modifyExtracellularQuantity(String id, double quantityPerTime){

    Substance ss = extracellularSubstances.get(id);
    if(ss==null){
            ss = ecm.substanceInstance(id);
            extracellularSubstances.put(id, ss);
    }
    double deltaQ = quantityPerTime*Param.SIMULATION_TIME_STEP;
    double volume = soNode.getVolume();
    degradate(ecm.getECMtime()); // make sure you are up-to-date weight/ degradation
    ss.updateQuantityBasedOnConcentration(volume); // TODO : is this step really necessary ?
```

```
        ss.changeQuantityFrom(deltaQ);
        ss.updateConcentrationBasedOnQuantity(volume);
        // we will diffuse next time step
        onTheSchedulerListForPhysicalNodes = true;


        }
```

%%

SimpleMorphogeneSecretor.java:

```
public void run() {
        PhysicalObject po = cellElement.getPhysical();

        // 1) Probe the concentration

        double A = po.getConvolvedConcentration(substanceA);
        double H = po.getConvolvedConcentration(substanceH);

        // 2) Compute the modification of A and H based on GIERER &
        // MEINHARDT

        double dA_dt = RhoA*A*A / H  + sigma_a;
        double dH_dt = RhoH*A*A + sigma_h;

        po.modifyExtracellularQuantity(substanceA, dA_dt*10);
        po.modifyExtracellularQuantity(substanceH, dH_dt*10);

      }
```

%%%

ExtendedSuperpositionSecretor.java

```
public void run() {

        PhysicalObject po = cellElement.getPhysical();

        // 1) Probe the concentration

        double A = po.getConvolvedConcentration(substanceA);
        double H = po.getConvolvedConcentration(substanceH);

        double Aother = 0;

        for (int i = 0; i< otherActivators.size(); i++) {
                Aother += po.getExtracellularConcentration(otherActivators.get(i));
        }
```

```
        // 2) Compute the modification of A and H based on GIERER &
        // MEINHARDT and competition with other activator substances

        double dA_dt = RhoA * A * A / H + sigma_a - rep_coeff * Aother;
        double dH_dt = RhoH * A * A + sigma_h;

        po.modifyExtracellularQuantity(substanceA, dA_dt * 10);
        po.modifyExtracellularQuantity(substanceH, dH_dt * 10);

}
```