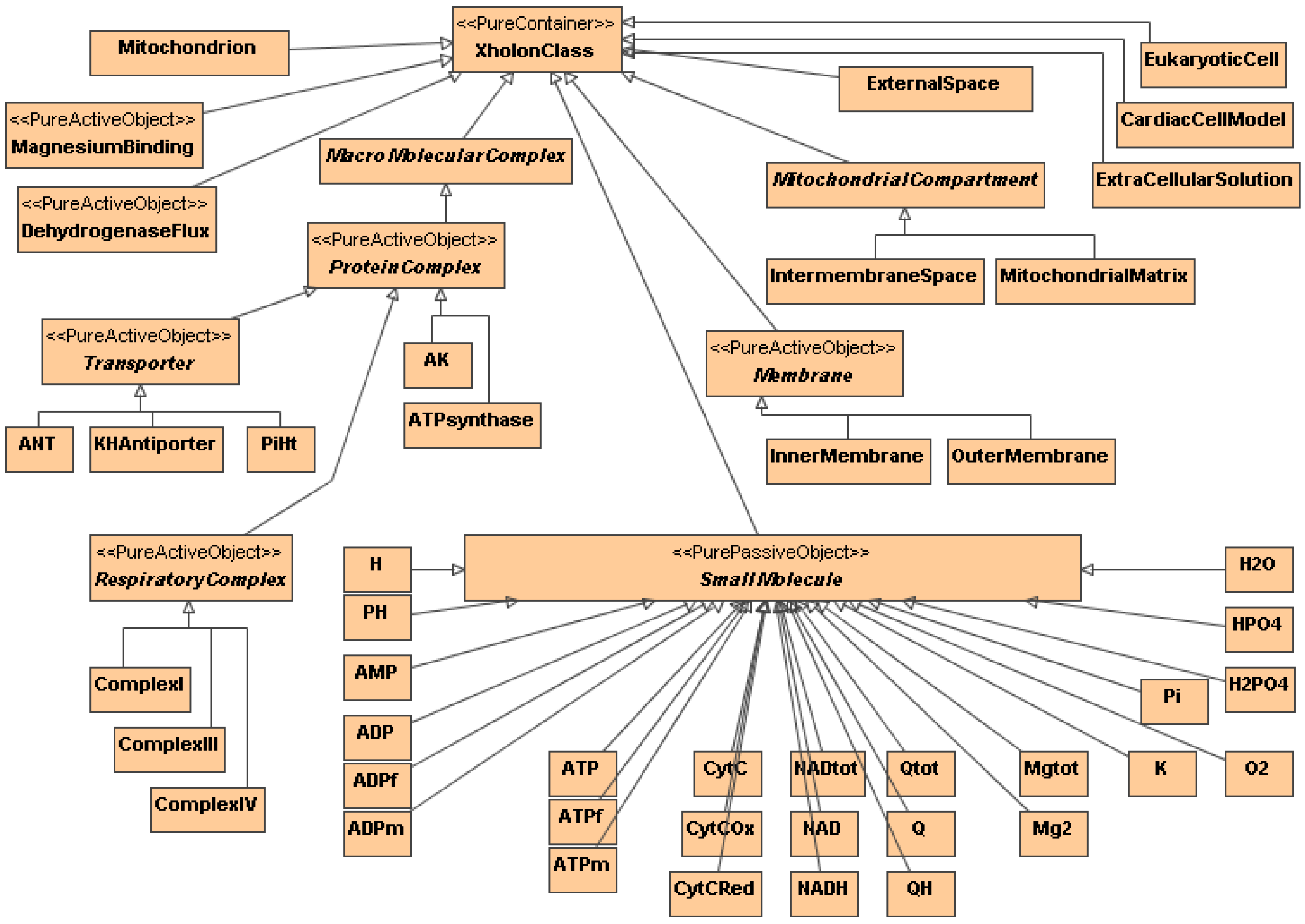
Mitochondrial Model – UML/Xholon – Step 4

Ken Webb
February 2, 2007

This is a brief description of an object-oriented model that I've created using MagicDraw UML and Xholon. It's intended to demonstrate the third and fourth steps of the CellAK process described in the BioSystems paper. This step includes both drawing of UML models and creating Java code.

An updated UML class diagram is shown below.



The third step in the BioSystems paper is as follows:

Step 3: Define external and internal behavior patterns
The third step adds behavior to the existing structure.
1. Define the desired behavior of the system by specifying patterns of message exchange between capsules [objects],
2. Define the detailed behavior of each capsule using state diagrams, the combined effect of which will produce this desired overall pattern of message exchange.

Neither of these sub-steps are used in the current Mitochondrial Model using UML and Xholon.This model does not use asynchronous message passing, but instead uses direct access by active objects to data values maintained by passive objects. This is mostly because Xholon has more options available to it than Rational Rose RealTime which was the tool used in the BioSystems paper.

The fourth step in the BioSystems paper is as follows:

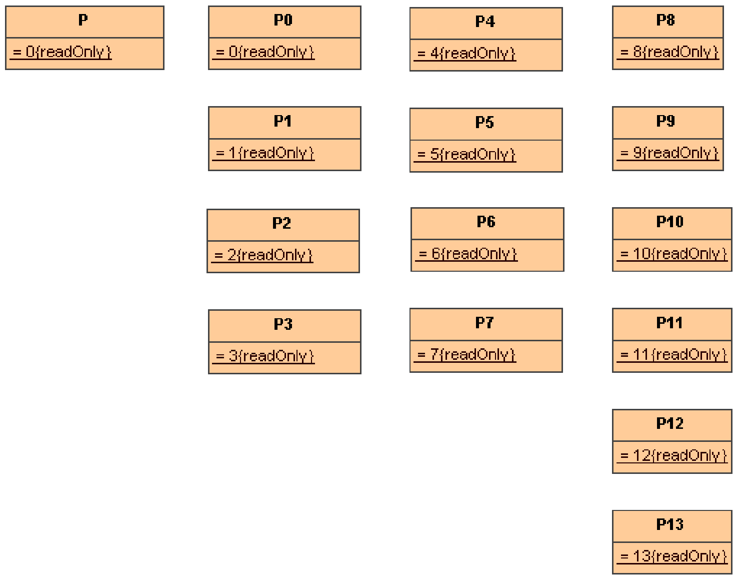
Step 3: Implement detailed behavior
.

Step 2.1

In this model, active objects such as enzymes, transporters, bilayers, and complexes interact with small molecules. Each small molecule can be accessed by any number of active objects.

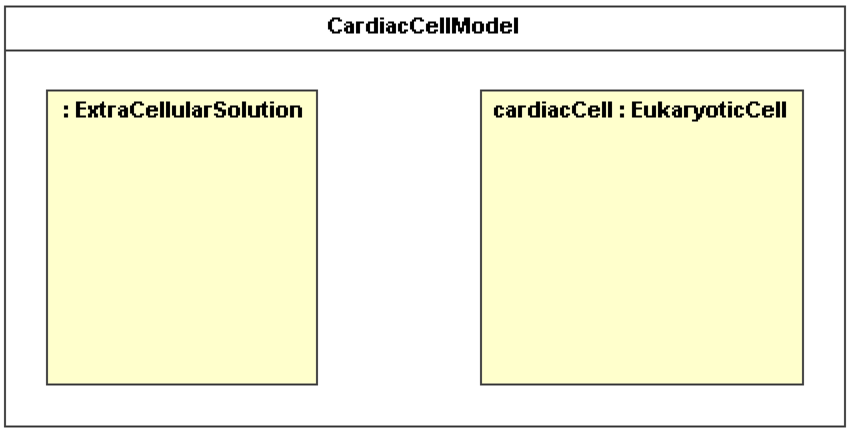
Step 2.2

A set of port classes has been created using MagicDraw. Each port class has a name and an identifying number. The same ID number can be used twice, but only if the duplicates are used for different objects. P will be used for small molecules, while the other 8 port types, P0, P1, to P13, will be used with active objects.

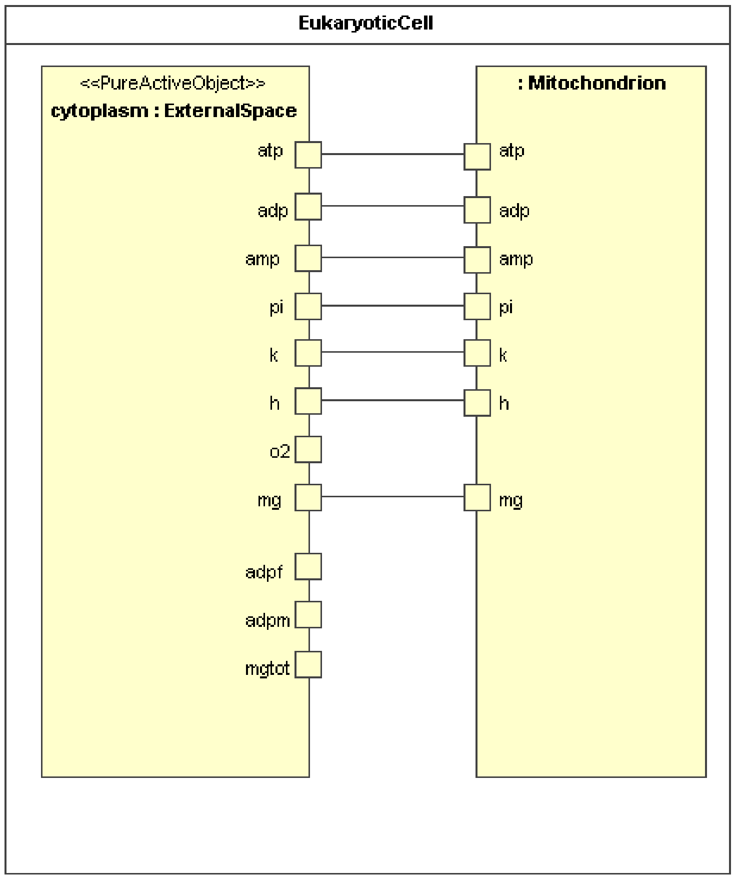


Step 2.3

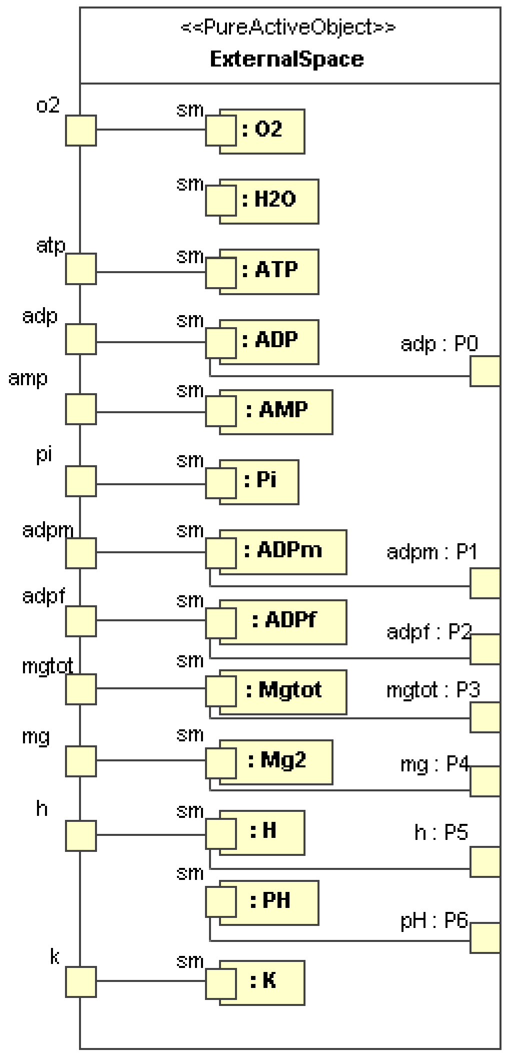
In step 2, I have added an extra level on top of the Mitochondrial Model, to make it easier to extend the model later if required. The following diagram shows the new top level of the composite structure hierarchy. In the CardiacCellModel, the EukaryoticCell is identified as having the role of cardiacCell.



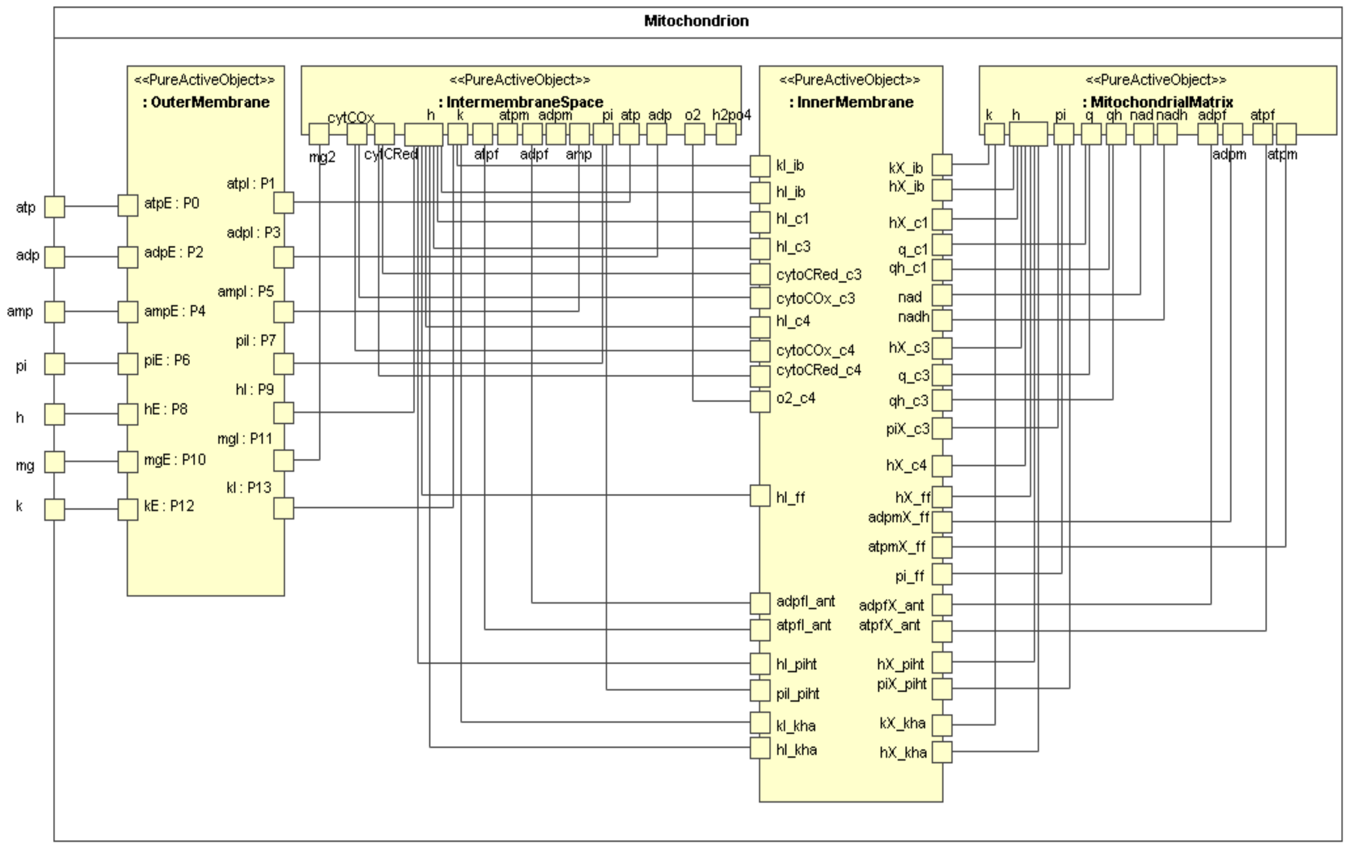
The next diagram shows the structure of the EukaryoticCell class. A Mitochondrion interacts with an ExternalSpace that plays the role of cytoplasm. The small squares along the edge of each object are port instances, labeled to identify the type of small molecules accessed through that port. For example, Mitochondrion can access ATP small molecules through its atp port.



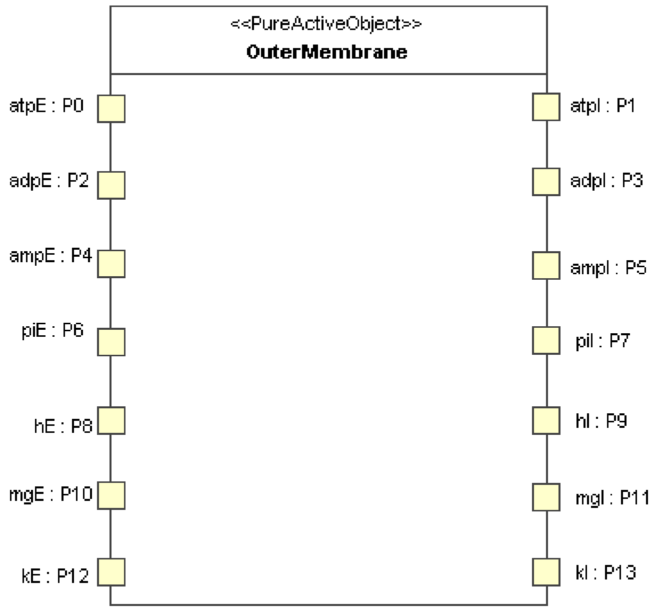
ExternalSpace is modeled as follows. Some of the ports are external public ports, while others are internal protected ports. The public ports, such as o2, atp, and adp, are visible to objects outside the ExternalSpace. The protected ports, such as adp:P0, are entirely within the ExternalSpace and are not visible to anything outside. These are used to connect the behavior of ExternalSpace to various small molecules inside the space. This will allow ExternalSpace to act on the small passive objects. Note that in this version of the model, ExternalSpace is an active object.



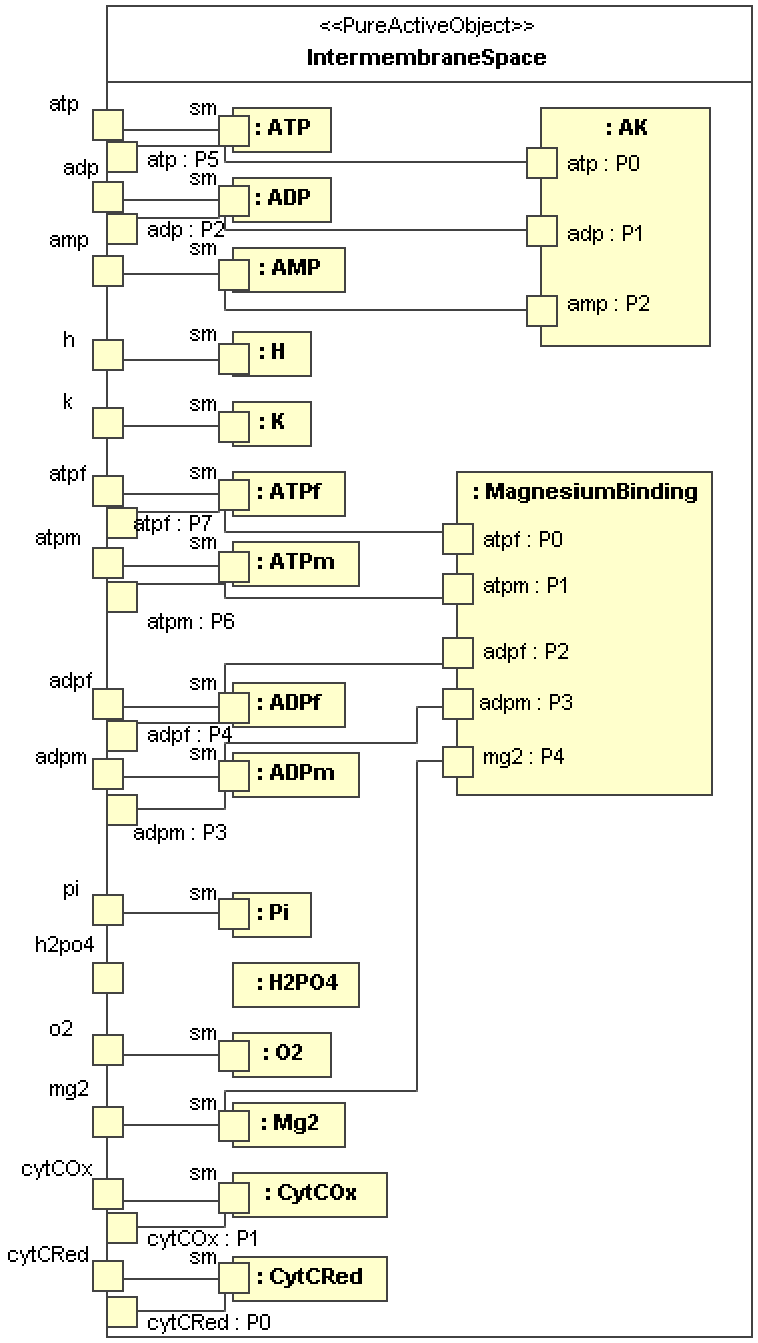
Mitochondrion has the following internal structure. Mitochondrion is just a container for four of the five major parts identified in the Beard paper – OuterMembrane, IntermembraneSpace, InnerMembrane, and MitochondrialMatrix. The fifth Beard part is ExternalSpace. Note that all five of these are now defined as active objects.



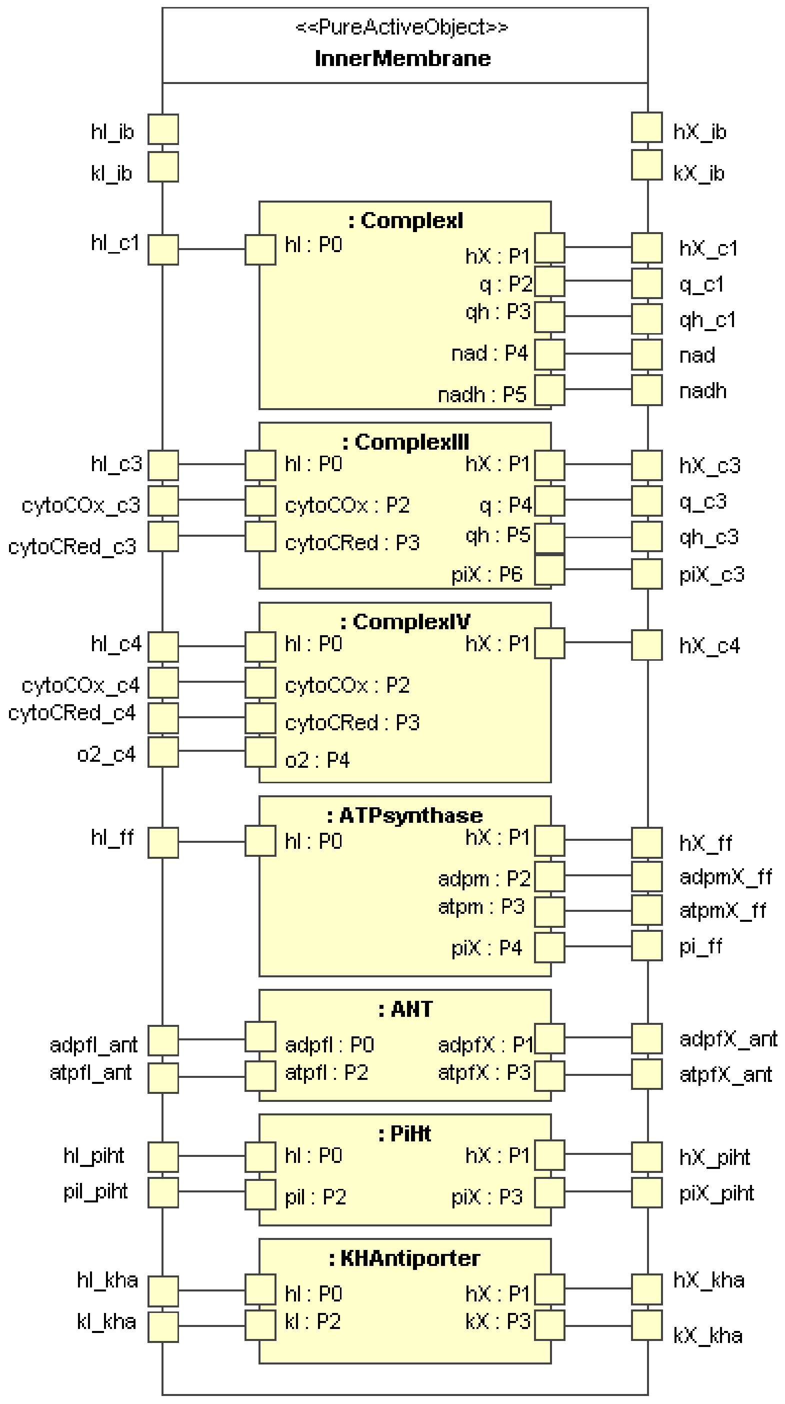
Each object shown in the above Mitochondrion diagram is itself a container. The following diagram shows the internal structure of OuterMembrane. It now directly contains the behavior that in the previous version was performed by OuterBilayer.



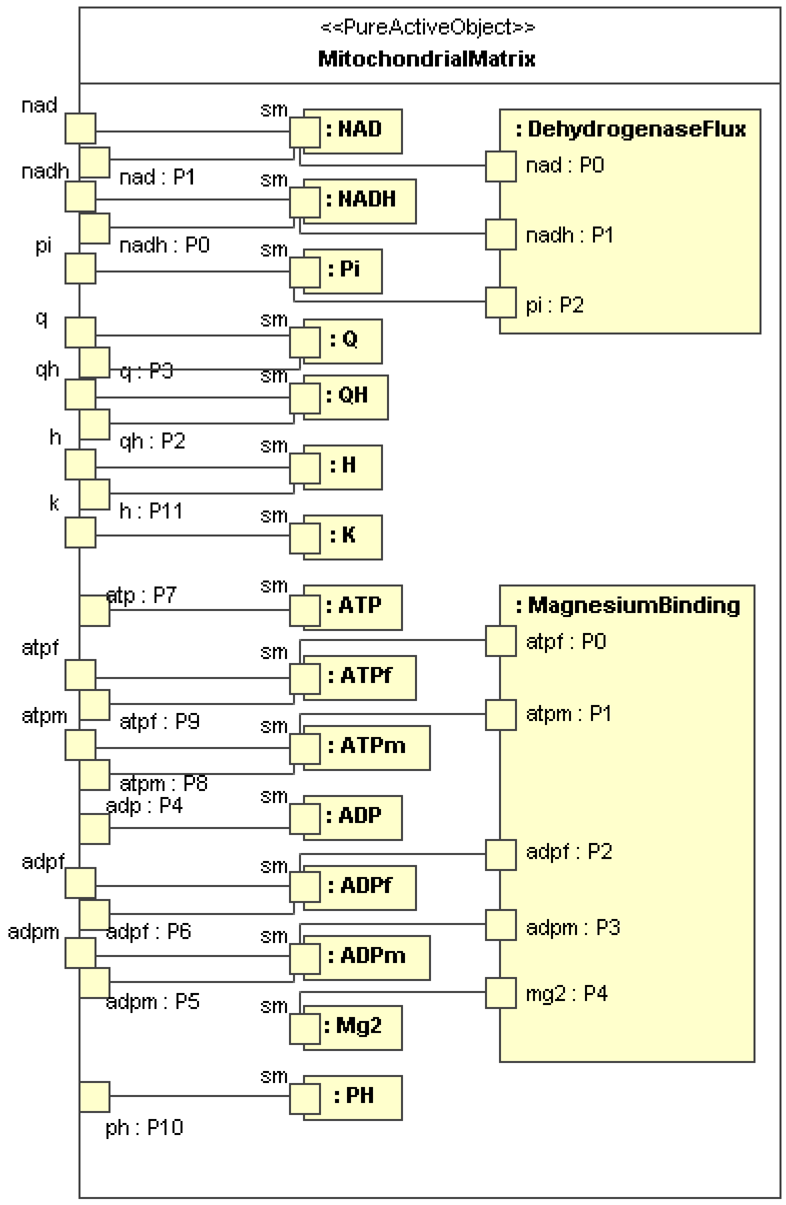
IntermembraneSpace is as follows. It contains 15 types of small molecules, all of which can be accessed by any number of active objects. These small molecules each contain a single value, which is how many instances there currently are of that type, for example, how many moles of Hydrogen (H) in the intermembrane space. IntermembraneSpace also contains two active objects, AK and MagnesiumBinding, that act on some of the same small molecules.



InnerMembrane has the following internal structure in this model. Each of these internal objects is an active object, as is InnerMembrane itself.



MitochondrialMatrix has the following internal structure.

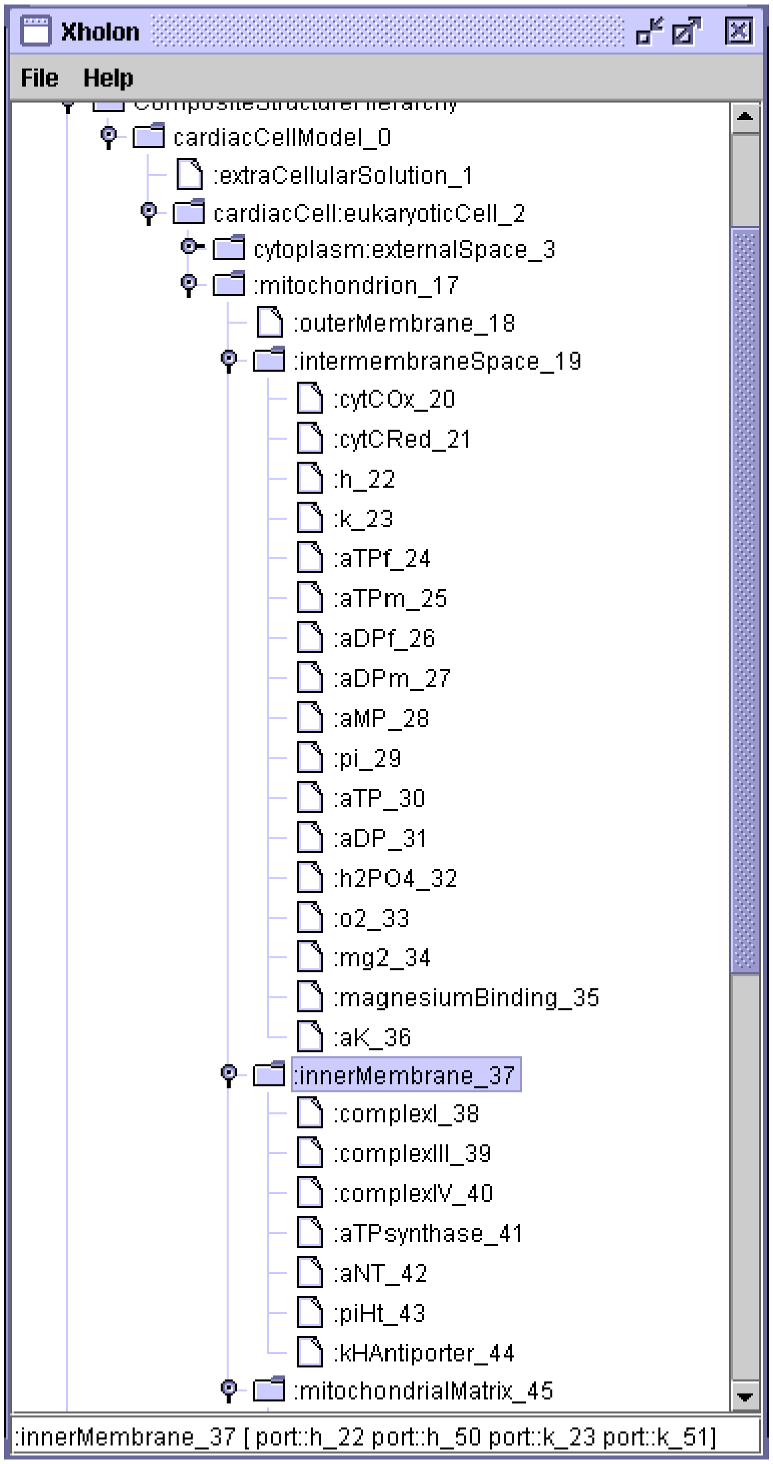


Step 2.4

The preceding diagrams also show the connectors between ports. These allow for actual interaction when the model is executed. Any number of connector lines can be drawn between an active object port and a port on a small molecule.

Runtime diagrams

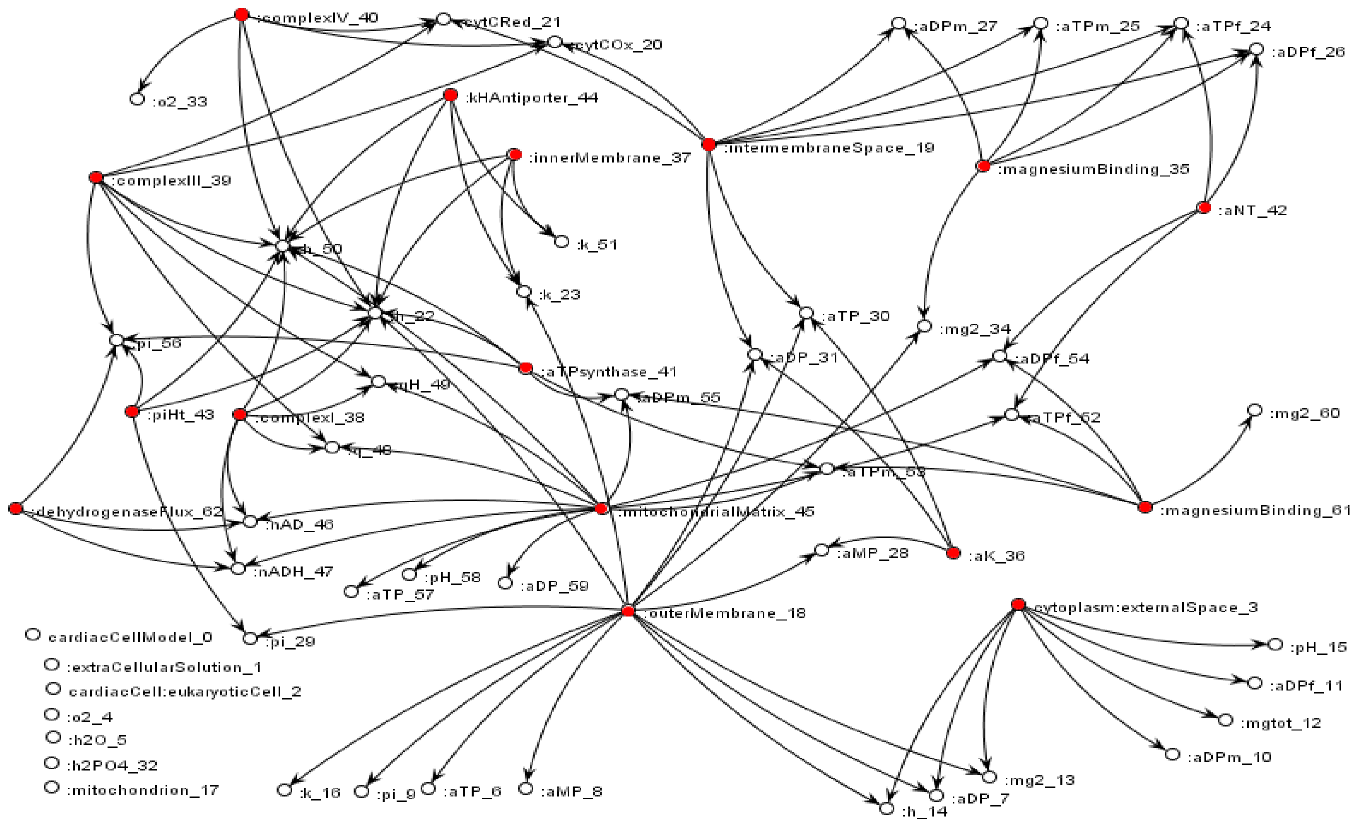
The following diagram shows the Xholon GUI when this model is run. The diagram shows the InnerMembrane selected, with four ports that it instantiates at runtime, so that it can actively move H and K between compartments.



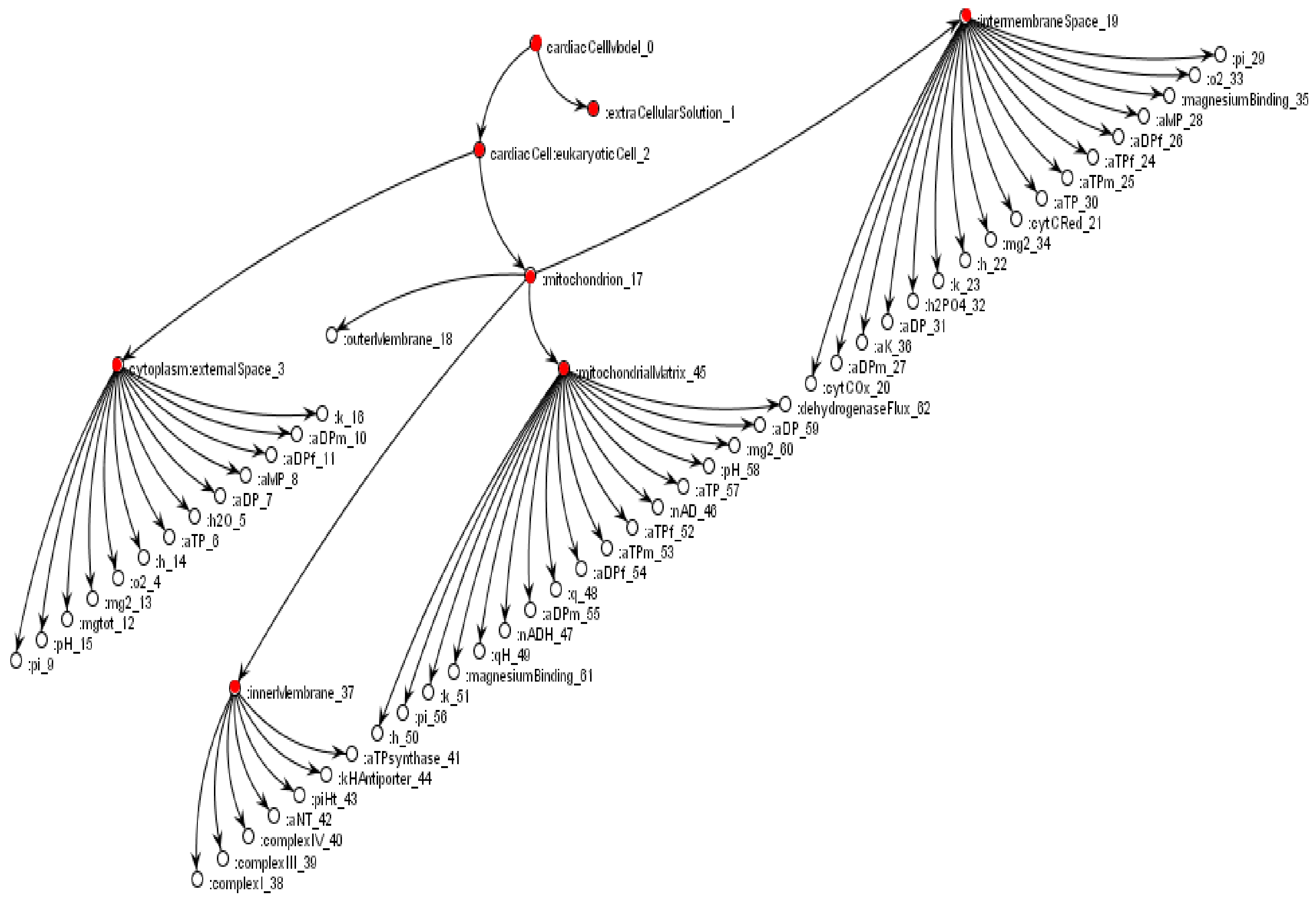
```
CompositeStructureHierarchy
  cardiacCellModel_0
    :extraCellularSolution_1
    cardiacCell:eukaryoticCell_2
      cytoplasm:externalSpace_3
      :mitochondrion_17
        :outerMembrane_18
        :intermembraneSpace_19
          :cytCOx_20
          :cytCRed_21
          :h_22
          :k_23
          :aTPf_24
          :aTPm_25
          :aDPf_26
          :aDPm_27
          :aMP_28
          :pi_29
          :aTP_30
          :aDP_31
          :h2PO4_32
          :o2_33
          :mg2_34
          :magnesiumBinding_35
          :aK_36
        :innerMembrane_37
          :complexI_38
          :complexIII_39
          :complexIV_40
          :aTPsynthase_41
          :aNT_42
          :piHt_43
          :kHAntiporter_44
        :mitochondrialMatrix_45
```

```
:innerMembrane_37 [ port::h_22 port::h_50 port::k_23 port::k_51]
```

Xholon can draw a variety of graphs, including the following biochemical network. I've identified the active objects in red, and have manually moved the nodes somewhat to improve the layout. Xholon uses the JUNG software to automatically generate this type of network graph. As an example, InnerMembrane is shown connecting to the same four small molecules as in the above Xholon GUI diagram.



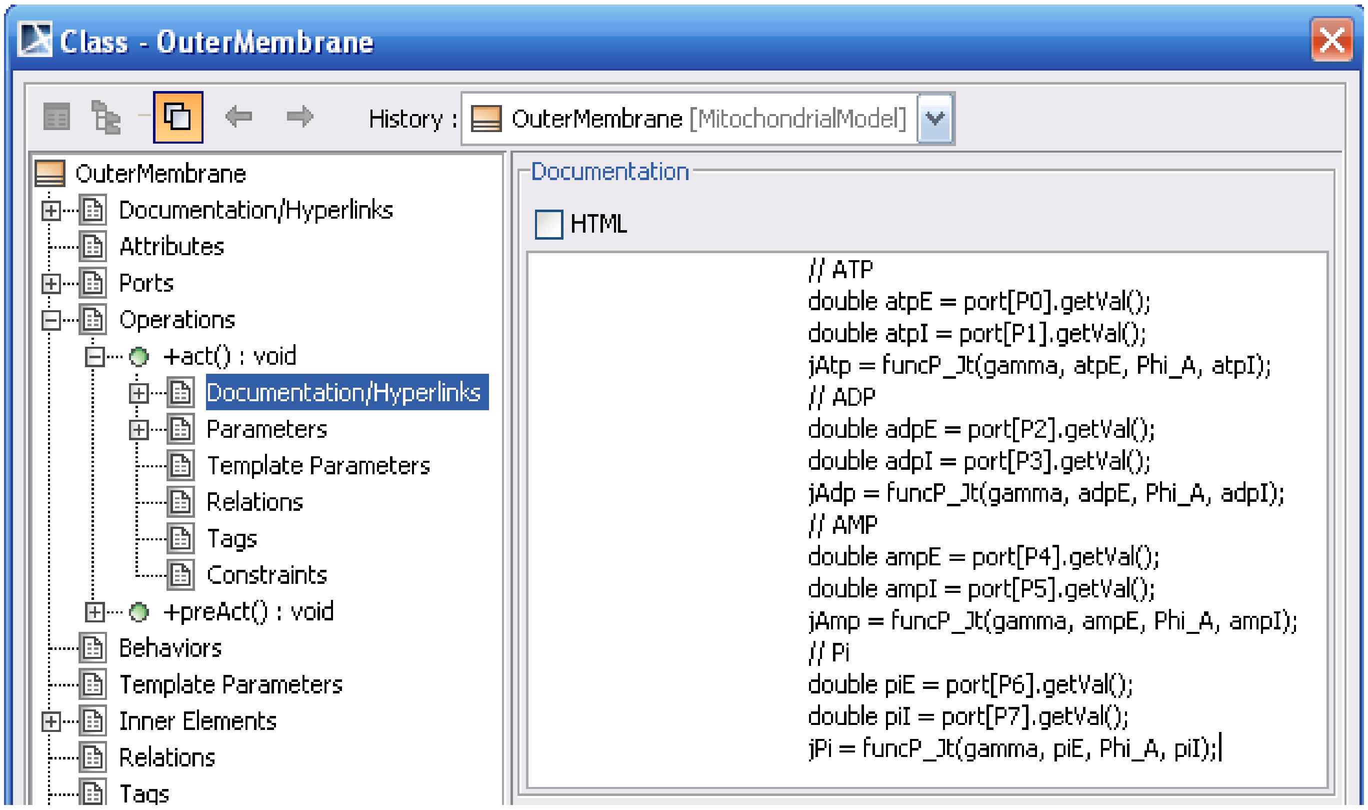
Xholon and JUNG can also generate a tree diagram such as the following, that shows the complete composite structure for the model. This is the same tree shown in the Xholon GUI diagram, as defined in the MagicDraw UML model..



Step 3 and Step 4

Step 3, as noted above, is not explicitly done in this model.

Step 4 involves implementing detailed behavior, which is where the bit of Java programming is required. At each time step, each active object needs to transform or move some small quantity of one or more small molecules. The behavior of OuterMembrane is shown in the following diagram. The diagram shows how this would be done using the MagicDraw UML tool. The membrane gets values of small molecules through its ports. It then does the same computation that is defined in the Beard Matlab model, and in the Simulink model. It does this by calling a Java function, for example by calling funcP_Jt.



```
// ATP
double atpE = port[P0].getVal();
double atpI = port[P1].getVal();
jAtp = funcP_Jt(gamma, atpE, Phi_A, atpI);
// ADP
double adpE = port[P2].getVal();
double adpI = port[P3].getVal();
jAdp = funcP_Jt(gamma, adpE, Phi_A, adpI);
// AMP
double ampE = port[P4].getVal();
double ampI = port[P5].getVal();
jAmp = funcP_Jt(gamma, ampE, Phi_A, ampI);
// Pi
double piE = port[P6].getVal();
double piI = port[P7].getVal();
jPi = funcP_Jt(gamma, piE, Phi_A, piI);
```