# PABS: Paradigm Application on Biological systems; a computational biological systems modelling tool

By Qurratulain Mubarak

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science
Leiden University
2007-2008

Supervisors:
Dr. ir. F.J. Verbeek
Dr. L.P.J. Groenewegen

Faculty: Leiden Institute of Advanced Computer Science
Date: 05/27/08
Version: 2.8

## Contents

## Table of figures

# 1 Summary

Model building can be an easier alternative to experimentation and models can be used for the purpose of process explanation and prediction. This thesis presents a different approach towards the modelling of biological systems than the traditional methods used. Traditional methods start directly from mathematical models rather than modelling at a more global level. These methods still lack the interaction dynamics. They do not offer a way of modelling the behaviour and the coordination between the systems consistently. By using a behavioural coordination language called Paradigm it has been explored how far it could facilitate the modelling of the behaviour, communication and collaboration taking place between the individual entities in a biological system. There is a huge demand for such a dynamic modelling technique, since such a dynamic modelling technique does not exist to date. Paradigm has been used for this purpose to model the interactions in a biological system. Some concepts from UML have also been applied in modelling and it has been researched which software is suitable for building dynamic models using Paradigm notations.

The Paradigm language can offer a new approach towards the modelling of interactions in living systems. Paradigm is a behavioural coordination language developed at the Leiden institute of advanced Computer Science. It has so far been applied in modelling business models or software components. The idea of using Paradigm in modelling biological systems therefore is new and has not been investigated so far. Two case studies from the biological field have been selected and modelled in this thesis. Both the case studies modelled, illustrate how Paradigm can be applied in modelling biological systems and how it can contribute to a deeper and better understanding of these systems.

# 2 Introduction

In biological systems the interesting and important aspects are the interactions between cells, genes, genetic networks, molecules or otherwise. To understand Biology at the system level, the structure and dynamics of the entire structure in separation of genes, cells, molecules etc must be examined rather than their individual characteristics. Properties of systems, such as its behaviour are an important aspect, and understanding these properties could have an impact on the future study of, for example medicine. Studying these interactions therefore requires a way of modelling them. To predict the behaviour of such systems and to model it there is a need for a modelling technique which can model all the characteristics of the entities such as its state, location and show the communication and collaboration taking place between the entities involved in a system. Such a technique does not exist so far which can model all these characteristics, since traditional methods start directly from mathematical models rather than modelling at a more global level. These methods still lack the interactions dynamics. They do not offer a way of modelling the behaviour and the coordination between the systems consistently. This has provided the foundation for the need of a new approach towards the understanding of complex biological systems. Bioinformaticians and systems biologists expect that building a good dynamic model of biochemical/genetic networks is a key step towards the development of predictive models for molecules or whole organisms. Such models are regarded as the keystones of Systems Biology. Therefore there is a huge demand for a dynamic modelling technique which has not been developed to date.

In this thesis I will present such a dynamic modelling technique which facilitates the modelling of the dynamic behaviour of biological systems. I will use Paradigm, which is a behavioural coordination language developed at the Leiden Institute of Advanced Computer Science for this purpose. I will explore how Paradigm can help in eliminating all the above mentioned obstacles and how it could benefit biological systems modelling. How Paradigm could offer a more dynamic approach and how it could provide a better understanding of biological systems will be investigated.

Since there is no software which can visualise Paradigm's concepts I researched about different software systems such as GenMAPP and Rational Rose. This was done to investigate which software is suitable for modelling using Paradigm as the coordination language. GenMAPP is standard software used by biologists to model biological systems and Rational Rose is standard software used by Computer Scientists for modelling of software systems. Paradigm's concepts were applied in both these software's and it has been investigated if a dynamic system could be built on top of them, which can visualize Paradigm's concepts.

I used some UML diagrams along with Paradigm notations. This approach I believe will offer greater potential modelling flexibility and power because of its use of the concepts of Object Orientation (UML) and Paradigm. The software development community has been using these concepts to build complex systems, and that level of complexity and experience can be used in systems found in biology. UML is starting to be used to a limited extent within the biology community. The Systems Biology Markup Language

(SBML) specification documents use many UML diagrams to formalize the SBML data structures. [1] The Object Orientation concepts of classes and inheritance can be of significant importance when building simulations of the systems in the future.

I will model two examples in this thesis. The first one is the binding of two molecules where they unite to form a bigger molecule. The other example is of a biochemical reaction where a protein acts as a catalyst in speeding up a reaction. In both these examples I will illustrate how by gradually adding more and more detail a system can be modelled using Paradigm and how it could provide a better understanding of these systems. This thesis deals mainly with the bottom-up behaviour of molecule binding, gene expression and enzymes and shows how such systems can be modelled using the approaches mentioned above.

The research proposal for this thesis can be found in the appendix with the details about the proposal. This thesis is organised as follows;

Chapter 3 deals with the methods and theories used in this thesis. The different concepts of Paradigm are explained in detail in this chapter. Descriptions about UML, GenMAPP and Rational Rose have also been given.

Chapter 4 is related to Biology, intended for those whom do not have a background in Biology so that they can understand the systems modelled.

Chapter 5 discusses the results. The two case studies have been modelled in this chapter and it is shown how Paradigm can be applied in modelling biological systems.

Chapter 6 is about the research proposal I submitted in the NWO Mozaiek programme. This chapter gives a good idea about the purpose and advantage of Paradigm in the biological field.

Chapter 8, this chapter summarises the main points discussed in the thesis and concludes them.

Chapter 10, this is the Appendix. The research proposal for this thesis can be found here.

# 3 Literature and background information

## 3.1 Paradigm

The coordination and behaviour of biological entities will be mapped using Paradigm notations, developed in Leiden University, the Netherlands, with the help of UML diagrams [2]. UML has only recently been used for modelling; Paradigm was initially inspired from Object Oriented/Simulation languages such as Simuli. GenMAPP which is a software used for biological systems modelling has been used to draw the diagrams in this chapter so that it could be researched if it can be extended into a dynamic software visualising Paradigm notations.

### 3.1.1 Introduction of Paradigm

#### 3.1.1.1 Coordination Languages

Coordination belongs to the key concepts of Computer Science. Software systems usually consist of many components, communicating with each other and with their environments. Coordination therefore is the consistent organisation of the communication and its effects, such that required cooperation between all components involved is established [1].In this chapter I will explain Paradigm notations in terms of coordination between software components, which is the technical cooperation. Later on I will explain the use of Paradigm in Biological processes, that type of cooperation is called organic in Paradigm's world. Paradigm can also be used to model organic cooperation. This type of cooperation is often not very strict, often negotiable and implicit. [3]

### 3.1.2 Models

"There are things and models of things, the latter being also things, but used in a special way" Chao, Y. R. (1960); *Logic, methodology and philosophy of science* (pp. 558-566). Before I start with explaining about Paradigm and its role in modelling biological systems, I will explain the use and benefit of model building. Models are intended to help us deal in various ways with a system of interest. Do models provide better insights into the system, how do they fulfil this role and why does working on the model have any relevance to the real system? It is easier to approach this by casting the role of modelling as part of the process of explanation and prediction described in the following diagram:

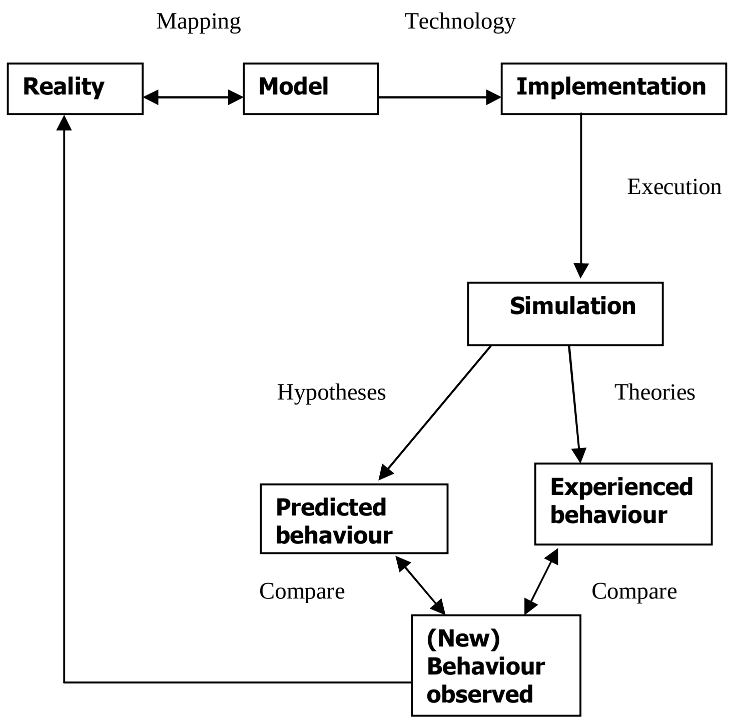


**Figure 1, Models and their use**

I will explain the role of modelling in terms of technology or rather a modelling language, since I will use a behavioural modelling language (Paradigm) to model the behaviour of biological systems. Using notions of certain phenomena you try to simulate behaviour of a system and create a link between that certain understanding or phenomena and theory. By doing this a link is created for explaining that phenomenon that is being observed and an effort is made to offer a better explanation for it. Using technology this same purpose can be achieved by simulating certain behaviour of a particular system. That simulation is either interpretated in predicting behaviour or that simulation is compared against some experienced behaviour that we already know. Therefore a model can either be used for prediction or comparison. Keeping that in mind, I will therefore try building a model which can either be used to predict biological phenomena or which can be used to compare against some experienced behaviour which is known from theories or experiments. Such a model can be used by biologists to test their hypotheses or theories. The need to perform experiments to understand behaviour of a system would be simplified, since such a model could give an overall impression of the behaviour of the system. Model building therefore can be an easier alternative, providing much promising results.

### 3.1.3 Concepts of Paradigm

Paradigm is a coordination specification language. Its name Paradigm is an acronym of PARallelism, its Analysis, Design and Implementation by a General Method. The important aspect of Coordination languages is the recognition of the relationships and coordination between the different components involved. With the help of such coordination languages used in system modelling, the entire system and its main processes are recognized. The entire dynamics of such a system can be identified in such a way. The same can be applied to biological processes where the dynamics of a biological process need to be modelled. Mostly Coordination languages such as UML and even Paradigm have so far been applied to modelling software world processes. The use of Coordination languages to biological systems is quite limited, while it could be investigated if it can be applied in an equally successful manner. Coordination and communication takes place in biological processes as well, maybe at a more detailed level, but using a Coordination language to model it should not be very different than modelling software components. In this thesis the possibilities of applying a coordination language i.e. Paradigm to biological systems modelling will be explored. There is no software yet for building diagrams using Paradigm's concepts. Therefore I used different Rational Rose and GenMAPP for this purpose where I built diagrams using Paradigm's concepts within the software. I will explain more about them later on.

In Paradigm an entire system and the components making up the system, behave in some cooperative or collaborative manner. The components can be viewed as collaborating between each other and a managing component that might be controlling/checking the communication between the components. The cooperation between the components can be of different types; excellent, counterproductive, non-existing, faltering or blocking. To achieve cooperation, the components involved communicate with each other by sending and receiving messages and signals. Sending and receiving can be of two types; Synchronous and Asynchronous. Synchronous is when sending and receiving is happening at the same time. Asynchronous is when a message is sent, the receiving can be done at a later time. The communication may occur between one to many components or one to one, where an object has a relation between many subobjects or to one subobject only. At the same time the sending component and the receiving component may be known to each other or not at all.

The goal of such communication always is to achieve cooperation so that the behaviour of the various components involved can be attuned. In Paradigm the goal always is to organise the coordination between involved components as efficiently and effectively as possible in view of the cooperation one wants to achieve. For this purpose in Paradigm the components can have a detailed level of behaviour and the global level of behaviour, through which they achieve effective coordination. This is another reason why I believe Paradigm can be applied successfully in modelling Biological processes.

In Paradigm the coordination of the components or agents is kept on a separate global level. This global level is composed from additional global levels of different component. In this way each component to be coordinated has its own local global level which is consistent with the detailed behaviour of the entire system. Therefore Paradigm has two globality levels; one for each separate component to be coordinated and the other for its consistent integration with the detailed behaviour.  This results in consistent dynamics of the components making up the entire system.

### 3.1.4 Concept of behaviour in Paradigm

In Paradigm the tasks that are relevant for what has to be coordinated are described first. This description is done by defining the basic tasks to be performed in sequence and the sequence describes the order in which the tasks are to be performed. Within the Paradigm framework such basic tasks are called a process, visualized as State Transition Diagrams (STD). Detailed behaviour in Paradigm model therefore is represented by a STD. A STD consists of a set of states, a set of transition labels or actions, and a set of transitions each linking two states by a transition label. The executed or realized sequence of steps from basic tasks is called behaviour. The detailed behaviour of an element in a Paradigm model is described with the help of all the states and all the transitions as the components move from one state to the other.

For the coordination between processes global behaviours are defined. Since STDs are the basic components for behaviour and coordination, the global behaviour is also derived from STDs. For the description of global behaviour of a process, Paradigm uses two additional concepts: subprocess and trap. A subprocess temporarily restricts the complete behaviour; it is a (behavioural) part of the process. It can also be seen as a phase in the complete behaviour and during this phase the basic tasks to be performed are restricted to certain situations or subset of transitions. The constraint introduced by the subprocess on the process is meant to be temporary and is imposed from the outside by the manager. Therefore each subprocess is described by a subSTD, which is a part of an STD. Subprocesses of one STD can overlap, a trap of a subprocess is a subset of the overlapping part of two successive subprocesses. A trap is defined as a subset of states in a subprocess, which once entered, cannot be left, unless proceeding to the next subprocess. It is a signal for a subprocess of being ‘ready’ to execute the next task. Traps can be nested; the trap that comprises all the states of a subprocess is called a trivial trap. A trap indicates a final stage of a subprocess. The global behaviour is then defined as a sequence of phases with phase changes in between which is similar to a sequence of states with state changes in between.

Paradigm uses some additional concepts such as Manager, Employee, Consistency rules and Partition. A Manager controls the overall behaviour of its employees and keeps a check on its phase changes. An employee describes the global behaviour in terms of phases (subprocesses) and phase transitions (traps). The collection of all the subprocesses is called a Partition of the STD. The sequence of subprocesses, arranged by traps from each subprocess to the next is controlled by so-called ‘consistency rules’. These concepts will be elaborated in the following sections as I introduce the formal notations of Paradigm later on.

### 3.1.5 Concept of Coordination in Paradigm

Coordination in Paradigm is formulated in terms of the combined global behaviours of the communicating components, i.e. in terms of combined behaviour of behaviour, and phase changes. This is a form of coupling on global levels only, and shows how other STDs will react on corresponding partitions. Manager and employee concept is thus introduced.

In Paradigm the coordination between a manager and its employees is of prime concern. This is achieved by relating the local behaviour of the manager with the global behaviour of the employees, so that consistency in the system is achieved. The employee process is not really subjected to the manager process; it is just to set up such a connection that shows how the employee process reacts when a manager process is in motion. The manager will check the current status of an employee constantly before it moves on to a next state, to know whether the employee has reached a certain trap that permits this transition. Only if the employee has reached that trap will both the manager and employee be able to move on, otherwise the manager will need to wait until the employee has reached such a trap that will allow it to move on. During some time interval an employee is in a behavioural phase, from which the employee moves to a next behavioural phase. This can only happen after a certain stage of the former phase has been reached. Where and when exactly in that stage the employees actually changes its phase, does not matter for the manager. The interaction between manager and employees is defined as consistency relation. How this relation is represented will be explained in the following section.

The manager process is usually chosen as the one that is keeping a check on the overall subprocesses and monitoring the phase changes, but there are no strict rules for choosing a manager process. The role of a manager and the role of an employee can be switched. A manger can become an employee of some other manager process in the overall process. The employee itself can be a manager of some other process. If the manager does not become an employee of some other manager process then it has no subprocesses.

### 3.1.6 Formal Notions of Paradigm: Syntax

#### 3.1.6.1 Syntax of Basic Concepts

1. In Paradigm, a *Process or STD* $P$ is a tuple, $P$ =<ST, AL, TS>.

ST is the set of states, which contains all the states an STD can have, belonging to an actor. AL is called the set of actions or transitions labels. TS is the set of labelled transitions, from one state to a new state, via a *transition label* $a$, is written as $x \xrightarrow{a} x'$, where $(x, a, x') \in TS$ .

2. A *subprocess* of a process $P$ =<ST, AL, TS>, is denoted as $S =< st, al, ts >, st \subseteq ST, al \subseteq AL, ts \subseteq TS$ a process.

The concept of subprocess is to divide a process STD into several stages, or temporary restrictions.

3. A *trap* $tr$ of a subprocess $S_i =< st, al, ts >$ is a nonempty set of states $tr \subset st$, with $x \in tr$ and $x \xrightarrow{a} x' \in ts$, implying $x' \in tr$.

A trap is included in a *subprocess*, usually has a smaller set of states than the subprocess has. If $tr = st$, then we say the trap is a trivial trap, as it contains all the states there are in the subprocess.

4. Trap tr is a trap of subprocess $S_i$, *tr* is called a *connecting trap* from subprocess $S_i$ to another process $S_j$, if the states in tr also belong to $S_j$.

5. A partition $\{(S_{i,}, TR_i) \mid i \in I\}$ of a process S = <ST, TS>is a set of subprocesses si = <sti, tsi> with traps ti such that ST = Ui∈I sti and TS = Ui∈I tsi.

A partition of a process S is denoted as $\pi_i = \{(S_{i,}, TR_i) \mid i \in I\}$, with $S_i$ subprocess of S and with $TR_i$ is a set of traps of $S_i$. $TR_i$ is the set of traps, $\{tr_1, tr_2, ...tr_n\}$. A partition contains subprocesses of a process STD concerning a particular function or purpose. A process can have multiple partitions, and a partition can have multiple subprocesses and a subprocess can have multiple traps.

6. McPAL: Managing changing Processes ad libitium. It is a component that allows for modifying the dynamics of the system while all components remain in execution in a dynamically consistent manner. The important concept here is that a process within a model is viewed as subprocess of an unknown larger process. This allows for defining new fragments by modelling them just in time. Via the JIT modelling McPal offers the possibility to extend the already existing Paradigm model with new behaviours for the various processes, while keeping the execution of the model constrained to the already existing model. McPal can on the basis of new added dynamics coordinate global level behaviour, leading to a new evolutionary phase for each component. [5]

### 3.1.7 Visual Notations of Paradigm

**Visual Notations:**
To make the textual representations more clear I will give here the visual representations of Paradigm notations. This should make Paradigm's notations more clear. These same visual representations will be used further along with some UML, and GenMAPP diagrams. I will give visual representations to better describe biological processes in the later parts of this thesis. The main aim is to apply Paradigm in modelling biological processes to make the behaviour and coordination of such systems clear, visual representations are very important. This should help in a new or different understanding of the processes and maybe even give a better understanding of the same process. The aim of this new understanding is to give a better understanding without changing the formal system.

**Process STD**

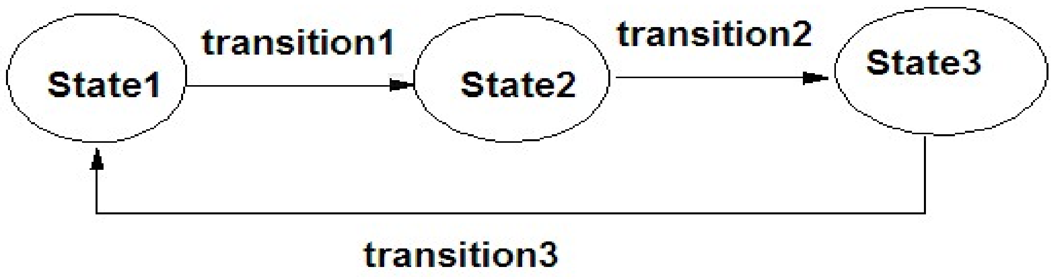


**Figure 2, Process STD**

A process STD will be represented as in figure 2. A state is represented by a round symbol with the name of the state written in. I have used round symbols since I think it is more in line with biological notations. A transition is represented by a directed edge, which connects two states, the beginning state is the starting state of the transition, while the state, which is connected by the arrow head is the ending state of the transition i.e. $x \xrightarrow{a} x'$. Often we see a loop here in the transitions as the states of an actor can be repeated.

**Subprocess and trap**

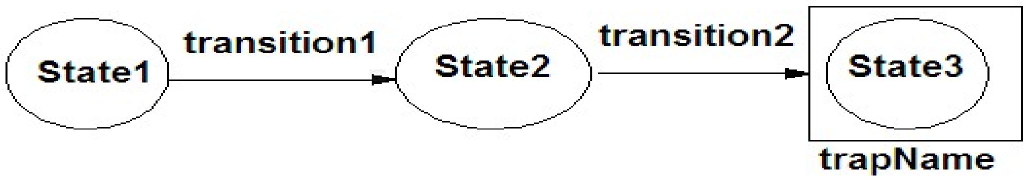


**Figure 3, Subprocess and a Trap**

A subprocess with a trap is shown as in figure 3. A subprocess is a subset of the process STD, which ends at the trap as presented in the figure. A trap is a state set that once the actor has entered, it cannot leave without further notifications have been instructed. A trap is represented by a polygon (rectangle) with solid border line, having the name of the trap written at the polygon’s border in small letters. The polygon with dashed border outside the subprocess represents a trivial trap.

**Global behaviour**
The global behaviour of a partition of a process is set up on top of the established subprocess structures. See figure 4.

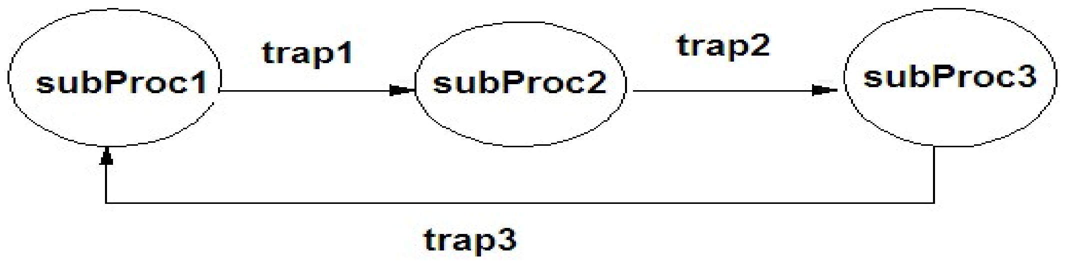


**Figure 4, Global Behaviour**

Every ‘state’ in the global behaviour figure shown above represents a subprocess from the same partition as illustrated in figure 3. The ‘transition edge’ represents a between trap as defined in Paradigm. A connecting trap can be trivial, which indicates globally the status of the subprocess remains unchanged. Typically, the global behaviour will be used in the managing behaviour illustration subjecting to consistency rules. It is the way how manager process manages its employees by controlling their progress only when necessary without interfering in the detailed behaviours. A global behaviour takes place on the basis of a particular partition containing a group of subprocesses.

**Manager and employee**
Managers and employees are interrelated and the state which *manager process* is in directly determines the state that an employee can be in.

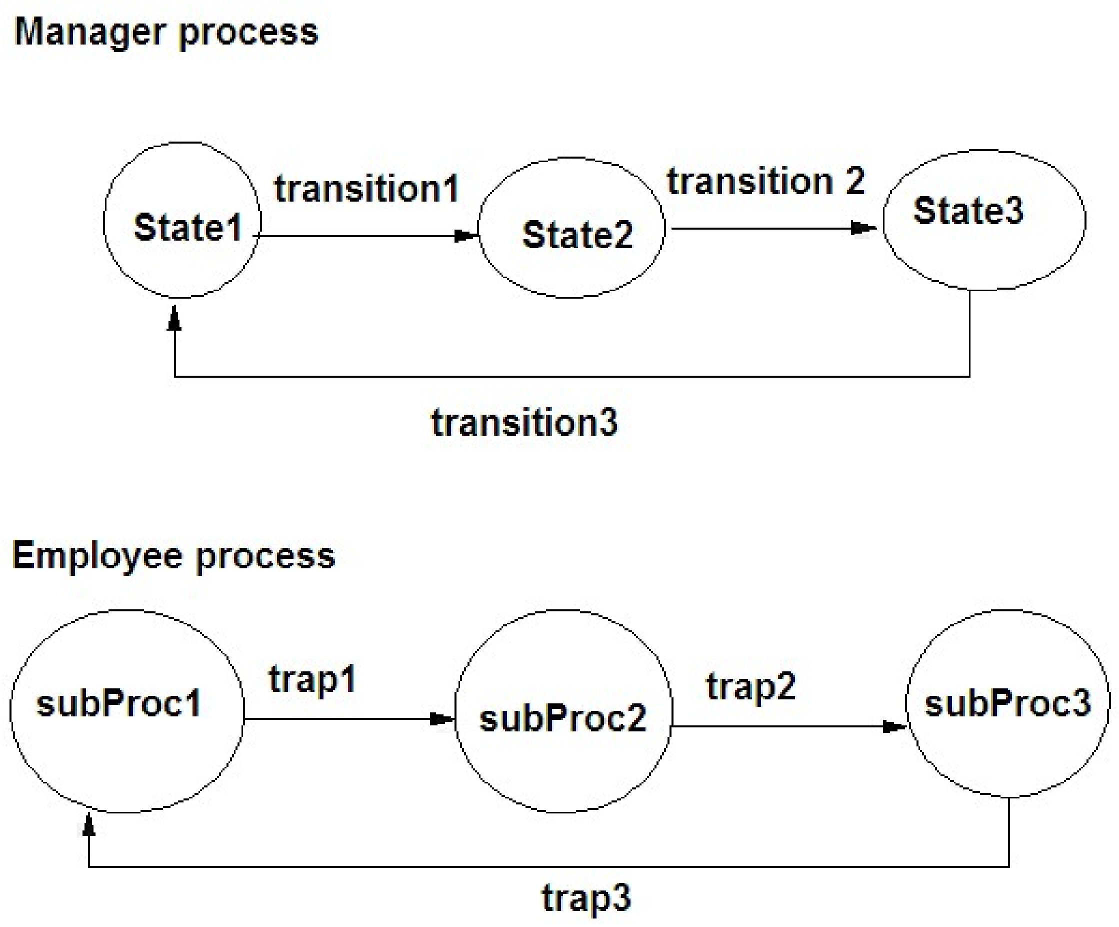


**Figure 5, Manager and Employee Processes**

As it is shown in figure 5, the visual representation for a managing process is presented. The figure shown maps the manager process and the employee process; the global variation of an employee process is mapped below a manager's local process, according to the running sequence. I have synchronised the employee's global process in the same manner simulating the manager process, in this way I have kept the behavioural consistency between managers and employees. When the manager moves forward to the next state, it will check the employees' current status, and adjust the employees' subprocess into the specified one as mapped in the figure. For instance, in figure 11, before the manager moves from state 1 to state 2, it will check the employee's status, if the employee process currently resides in subprocess 1, and has reached trap 1, it will be able to enter state 2 by adjusting the employee's status to subprocess 2 as well. If the employee has not entered trap1, the manager cannot proceed to its next state, as it will need to wait for the employee to reach the connecting trap (trap1) of subprocess 1 and subprocess 2 of employee process. The global process gives the dynamics of the validity of constraints. Manager process mirrors this by mapping as detailed process, but it also mirrors it for other global processes.

The visual representation then provides, per manager state, the current coordination of its employee's subprocess, one for each employee.

Moreover, as we have a manager that manages several participating STDs (employees), the manager process manages their employee processes via the interface of employees, which in the course of time shows the global behaviours of the employees.

### 3.1.8 Consistency rules

Consistency rules are specified along with every managing figure, controlling and restricting the behaviour of the coordination between managers and its employees. These rules specify how the transitions and phase changes take place. They can look like the following;

$$
\begin{aligned}
&\mathtt{ProcP} : \mathtt{state\_a} \rightarrow \mathtt{state\_b}\ *\\
&\qquad \mathtt{ProcQ}_1[\mathtt{PART}_1] : \mathtt{SubProc}_1 \rightarrow \mathtt{SubProc}'_1,\\
&\qquad\quad \dots\\
&\qquad \mathtt{ProcQ}_n[\mathtt{PART}_n] : \mathtt{SubProc}_n \rightarrow \mathtt{SubProc}'_n
\end{aligned}
$$

Here state_a→ state_b is a ProcP transition, PARTi is a partition of process ProcQi and SubProci → SubProc'I is a transition in the global behaviour. Via a consistency rule, a combined transition occurs consisting of a state transition and zero or more subprocess. In the presence of the consistency rule the process ProcP is called manager of the processes. The latter processes are called employees of ProcP. So, an employee has at least one partition and, therefore, global behaviour.

The consistency rules for the manager shown in figure 11 would be as follows. The consistency rules are integrated into the STD of the manager process. For instance, the coordination when manager moves from state1 to state 2 can be equally explained in the consistency rule as shown below.

*Manager* (process): state 1 $\rightarrow$ state 2

Employee 1[partition 1]: subprocess 1 $\overset{trap1}{\rightarrow}$ subprocess 2

## 3.2 UML

Starting in the late 1980s various individuals and software development communities developed their own graphics-based methods for object-oriented analysis and design. In the mid 1990s Grady Booch, Jim Rumbaugh, and Ivar Jacobson merged their slightly different approaches into a common Unified Modelling Language (UML). The UML standardization process is managed by the Object Management Group (OMG). It is standard practice in the computer industry to present analysis, design, and implementation models of a system, using the UML common visual notations. UML has offered different views of a system, including functionality view, scenario-oriented, local behaviour and activity diagrams.

In the initial stages Socca and OMT was used to build Paradigm models. UML prior to 2.0 was not really suitable for this. After the new version 2.0 UML is quite suitable for building Paradigm models.

Normally the UML and other system specification languages provide different views and perspectives in representing an object, which enable the system developers a full-package of the internal design of the system. However this can sometimes cause confusion especially when you want to develop the relationships and connections between the different views. The behavioural consistency between the various components of a UML model still remains an open issue [2]. In the following chapter I will use some concepts of UML together with Paradigm to model a biological process and its behaviour and show how Paradigm can provide the behavioural consistency between the different views which UML is lacking.

UML has successfully been applied in modelling biological processes and is starting to be used to a limited extent within the biology community. The Systems Biology Markup Language (SBML) specification documents use many UML diagrams to formalize the SBML data structures. [1].That level of experience can be taken together with Paradigm therefore; I used some UML diagrams along with Paradigm notations. This approach offered greater potential modelling flexibility and power because of its use of the different concepts of UML, Object Orientation and Paradigm together.

**UML notations:** Basic UML notations used is the following type of figure. I used Rational Rose for building the following UML diagram. Rational Rose is standard software used by Computer Scientists to build system models. The diagram shown is a Collaboration diagram representing the collaboration between two entities.

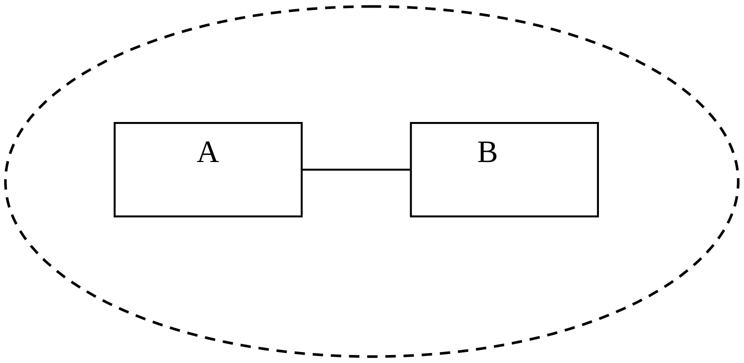


**Figure 6, Collaboration diagram**

## 3.3 GenMAPP

Along with UML based software I will also make use of a program called GenMAPP to visualize Paradigm’s concepts. This is for the purpose of experimenting with biological software to see if Paradigm concepts can be applied successfully in such biological software and to research if this software can be extended to a dynamic software system visualizing Paradigm notations. GenMAPP is an academically based organization that develops and supports GenMAPP (Gene Map Annotator and Pathway Profiler), a computer application designed to visualize gene expression data on maps representing biological pathways and groupings of genes [4]. It is standard software used by biologists to represent biological pathways. The following diagrams in this chapter have been developed in GenMAPP.

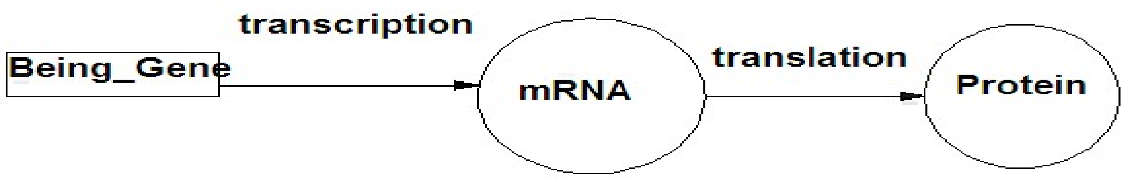


**Figure 7, STD using GenMAPP**

**GenMAPP diagram types:**
Process; I will represent a process state in an oval shape in this thesis with a capital naming. Actions and their labels are represented in small letters, such as in figure 7 above.

# 4 Basic Biological terms

In this chapter I will explain some basic biological terms used in this thesis so that the biological examples used for modelling will be comprehendible to people with limited biological knowledge. Starting from the very basics of Biology, I will gradually explain in more detail the structural details of the basic elements. Before giving definitions each time, I will explain about the biological terms used. In this way I hope the terms used will be easier to understand.

## 4.1 Cell

### 4.1.1 Cell Basics

Biologists call living things organisms. Organisms are mostly very complicated and highly organised, consisting of very small entities called cells. Cells are the fundamental units of life and all living creatures consist of cells. Cells contain very intricate internal structures each having their own specific function or purpose. There are two primary types of cells; the Eukaryotic cell and the Prokaryotic cell. Cells that contain a true nucleus are called Eukaryotic cells and those that do not contain a true nucleus are called Prokaryotic cells. A cell nucleus contains various organelles. Organelles are one of the intricate structures found in cells and can be seen as organs within the cell performing certain functions and having a specific purpose. Cells need energy to survive like all living things. Therefore they can extract energy from their environment which they use to maintain their structures. That energy can be derived in two ways according to the organism. Chloroplasts and mitochondrion are two different types of organelles found in cells which are able to produce such energy. Chloroplasts are found in plant cells and they are responsible for converting sunlight into energy for the plants. Similarly mitochondria are found in animals and they also provide energy to the cells. Here are a few definitions about cells.

**Cell**: All living organisms are built up from cells. Cells are organised in tissues and organs. Cells consist of cytoplasm, a cell membrane and organelles.
**Cell nucleus**: The organelle found in most Eukaryotes. The cell nucleus contains organelles and chromatin (consisting of DNA and proteins). What exactly chromatin is will be explained later, for now it can be seen as containing the genetic material of the cell.
**Prokaryotes:** Cellular organisms without a true nucleus, their genetic material floats in the cytoplasm.
**Eukaryotes:** Cellular organisms with a nucleus. The Eukaryotes comprise: animals, plants, and fungi.

### 4.1.2 Cell Detail

Cells do not only perform external functions such as extracting energy from their environment but perform various other internal purposeful functions as well. The most unique thing about living organisms is their ability to reproduce and replicate. They can reproduce and replicate almost perfect copies of themselves. Cells can replicate not just once or twice but for thousands of generations. To reproduce and replicate the genetic information present in the cell has to be transmitted. For this purpose cells have various organelles within them that help fulfil these tasks. The process of cell reproduction has three major parts. The first part of cell reproduction involves the replication of the parental cell's DNA. The second major issue is the separation of the duplicated DNA into two equally sized groups of chromosomes. The third major aspect of cell reproduction is the physical division of entire cells. The purpose and functions of the DNA and chromosomes is explained below.

Deoxyribonucleic acid (DNA) is found in the nucleus and it contains the genetic instructions used in the development and functioning of all known living organisms. Genetic information contained in the cell is encoded in the form of DNA. All cells contain DNA as their genetic material. DNA can be seen as the vast chemical information database that carries the complete set of instructions for constructing other components of cells, such as proteins and RNA cells. Proteins are essential parts of organisms and participate in every process within cells such as cell adhesion and the cell cycle. Many proteins are biological catalysts that speed up biochemical reactions and are vital to metabolism. RNA stands for Ribonucleic acid. It plays an essential role in the process of making proteins in the cell. Another form of RNA is mRNA. MRNA is RNA that carries information from DNA during transcription to the protein to undergo translation and create a gene product.

A gene can be defined as a region of DNA that controls a hereditary characteristic. Genes are the working subunits of DNA; they contain instructions for creating proteins and RNA cells. Within cells, DNA is organized into structures called chromosomes. These chromosomes are duplicated before cells divide, in a process called DNA replication. Chromosomes are organized structures of DNA and proteins that are found in cells. Chromosomes are vital for the health and growth of the cell. In resting state when cells are not dividing or reproducing, cells have chromatin which is made up of DNA, RNA and nuclear proteins. Once a cell divides the Chromatin becomes very compact and you see the chromosomes. Chromosomes also contain DNA-bound proteins, which serve to package the DNA and control its functions. All the genetic information contained in the cell is encoded in the DNA which contains all the instructions for cell reproduction and replication and all this genetic information is stored in the Chromosomes.

**DNA:** Vast chemical information database that carries the complete set of instructions for making all the proteins a cell will ever read. These instructions comprise the nucleotides A, G, C and T, adenine, cytosine, thymidine and gnamine.
**Proteins:** Any chain of amino acids. A gene’s DNA sequence is converted into mRNA which is translated into a protein. That protein is further used to perform certain biological functions.
**Chromatin**: The complex of DNA and proteins that build up chromosomes.
**Chromosomes**: Carry all the information used to help a cell grow, thrive and reproduce. Chromosomes are made of DNA.
**Gene**: Working subunits of DNA. Contain instructions, usually coding for a particular process.

## 4.2 Expression

Proteins are needed for energy transfer and for performing the organic reactions. These proteins are produced from genes in the cell. To understand the working of a gene it is expressed so that its instructions can be read out which are needed for performing the reactions. Expression therefore, is the process of converting a gene's DNA sequence into the structures and functions of a cell. During reactions of the cell, the genes within it are expressed. The following definition is taken from Wikipedia; " Gene expression, or simply expression, is the process by which the inheritable information which comprises a gene, such as the DNA sequence, is made manifest as a physical and biologically functional gene product, such as protein or RNA." The gene that is expressed can be used to give the cell control and structure or it can even have effect on its functions. Therefore this is an important process within the cell.

**Expression:** Process by which a gene's DNA sequence is converted into the structures and functions of a cell

The process of expression is carried out in two stages. In the first stage the DNA is transformed into RNA. That process is called Transcription. In the second process the RNA is transformed into a protein, this process is called Translation. That protein can be used in some biological function. The following figure visualises this process.

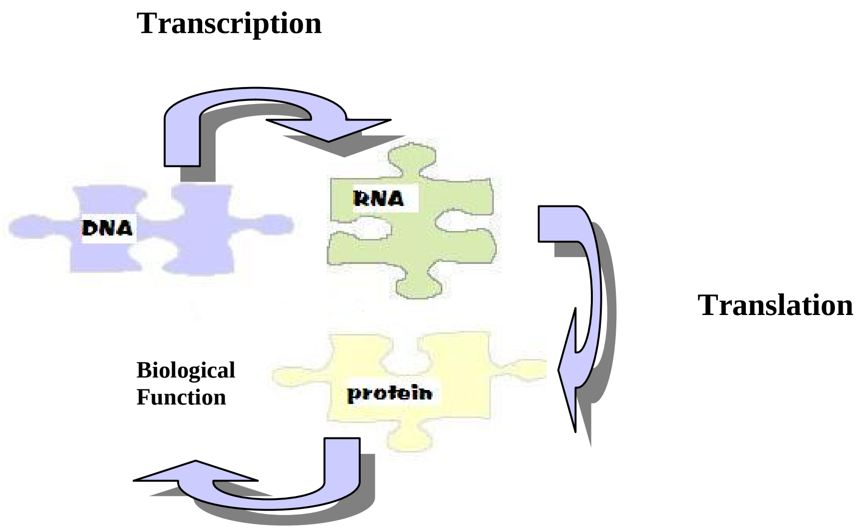


**Figure 8, Process of gene expression**

## 4.3 Catalysts

The reaction that I have modelled in chapter 5 is where a catalyst (enzyme) is involved in speeding up a reaction. A catalyst is a substance which speeds up reactions without itself being changed or consumed in the process. Enzymes are biological catalysts.

**Catalyst**: Substance that increases the rate of a chemical reaction without being consumed in the process. For a reaction to take place, the reactants must possess a certain amount of energy called the transition state at which they have enough energy to react with each other and form compounds. The energy of activation is therefore the amount of energy required to bring the reactants to that level of transition at which they can react with each other to form compounds. Catalysts combine transiently with the reactants to produce a transition state having a lower energy of activation than the transition state of the uncatalyzed reaction. Therefore they accelerate the chemical reactions by lowering their energy of activation. When the reaction products are formed, the free catalyst is regenerated.

**Enzyme:** Biological catalyst**,** mainly they are proteins**.** Enzymes are highly specific for a particular reaction. Purpose is to allow the cell to carry out reactions very quickly. They are made up of amino acids which are proteins and posses an enormous catalytic power. Their most specific attribute is that they act only on certain amount of substrates and only a single type of reaction takes place without side reactions or by products.

A series of enzymatic reactions is known as a pathway. The reactant that gets consumed in the reaction is called a substrate and the product is what is produced as the outcome of the reaction. These terms and the ones mentioned further are not explicitly used but they will help in understanding the biological process in chapter 5.

## 4.4 Biological terminology

**Coenzyme:** They enhance the activity of an enzyme
**Substrate :**( reactant) Gets consumed in a reaction.
**Product:** Produced by a reaction, for example;
H + O2 → H2O
(Hydrogen) + (Oxygen) → Water
**Pathway:** An ordered sequence of proteins and substrates. Can also be;
• A series of biochemical reactions
• An evolutionary process
• A biological system (living cell)
• A biochemical network/gap

Figure 9 clarifies what a pathway is. In the figure enzyme A and enzyme B are involved in a certain reaction. If we start reading from enzyme A then 1, 2 and 3 are first united. After the reaction, 3 gets separated and 1, 2 are left. Similarly 1 and 2 further react with enzyme B where they both are separated. This biological pathway can be read starting from enzyme B in the same way as starting from enzyme A.

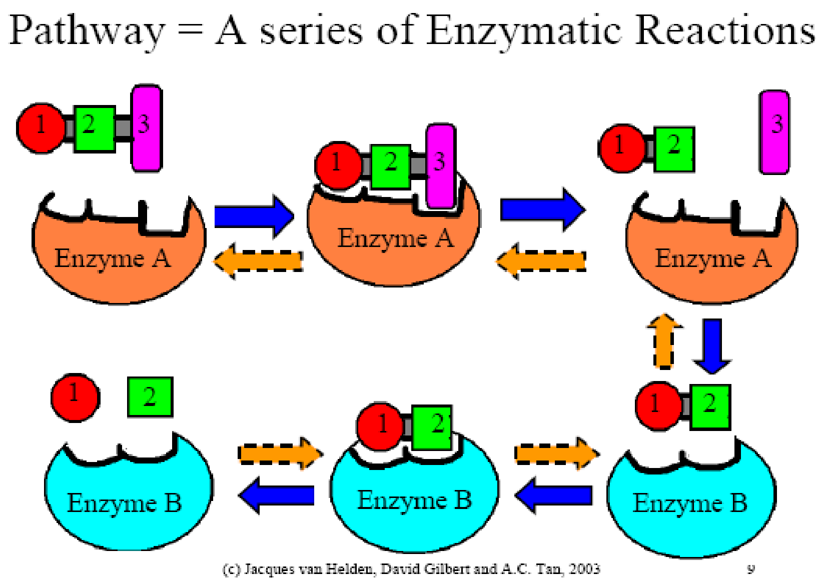

Pathway = A series of Enzymatic Reactions
1
2
3
Enzyme A
Enzyme B
(c) Jacques van Helden, David Gilbert and A.C. Tan, 2003
9

**Figure 9, Biological pathway [6]**

Biochemical reactions are classified according to the EC classification. EC stands for Enzyme Commission which is a standard used to classify enzymes according to the type of reaction catalyzed. Four numbers are used to denote the type of reaction for e.g. 2.7.3.2. The first digit stands for the class name, the second for the subclass, third for the sub-subclass and the last one designates the enzyme. Each reaction type therefore has a unique code and the type of reaction can be deduced from its E.C. number.

**EC Classification (EC):** Classified according to Enzyme Nomenclature (IUBMB) according to six major biochemical reactions**.** EC classification is denoted in four figures (EC X.X.X.X), according to the reaction taking place.

Although this is not directly a part of Biology rather more of biochemistry but I would like to name two chemical elements here; ATP and ADP. They are used in the biochemical reaction in chapter 5, so it’s useful to know more about them. ATP stands for adenosine triphosphate (ATP). ATP is the carrier of chemical energy in cells of all living things. When ATP transfers its energy to other cells it becomes adenosine diphosphate (ADP). ADP is the energy less form of ATP, but if it regains energy again in some form it can become ATP again.

# 5 Results

## 5.1 Bio-Paradigm

### 5.1.1 Introduction

In biological systems the interesting and important aspects are the interactions between their components. Such components exist at different levels of description and understanding such as cells, genes, genetic networks, molecules, tissues and others. To understand biology at the system level, both static and dynamics of the entire composite structure have to be investigated. This has to be done in terms of the various components across the relevant levels, rather than in terms of static or dynamic characteristics of individual components or at just one description level. Properties of systems, such as their behaviour are an important aspect, and understanding these properties could have an impact on the future study of biomedical research. Increasingly, there is a strong understanding by biologists that the behaviour of an individual component in a system is determined by its internal characteristics such as its state, its location and its relationships with other components in its environment. There is communication and collaboration between all these entities. The important part in each system is therefore to identify the components and the levels to be modelled and to integrate their separate descriptions on the basis of their interactions in a consistent manner. This then should result in a structural as well as behavioural description of the biological system as a whole. On the basis of such consistent integration across the various levels, of behaviour and interaction in particular, one can understand its overall behaviour. This requires suitable modelling of the relevant behaviours and most importantly, interactions.

Researchers in Bioinformatics and Systems Biology are increasingly using computer models and simulations to understand complex inter- and intra-cellular processes. The Systems Biology community is looking at alternative methods for modelling and simulating cellular and biological processes than the ones traditionally used. Traditional methods used lack the techniques for modelling the dynamic behaviour of systems. They start directly from mathematical models rather than modelling at a more global level. These methods still lack the interactions dynamics. They do not offer a way of modelling the behaviour and the coordination between the systems consistently. Lately there has been a strong realisation by the Systems Biology community for a method which can not only model the structure but also the behaviour of the system. Therefore there is a huge demand for a dynamic modelling technique.

This chapter will explore how Paradigm can offer a more dynamic approach towards biological systems modelling and what role it could play. Paradigm is a behavioural coordination language introducing phase dynamic on top of detailed behaviour. Paradigm is been developed at the Leiden Institute of Advanced Computer Science, for more detailed explanation about Paradigm see chapter 3. By using Paradigm research could be done on how Paradigm can offer a more dynamic approach and how it can help in eliminating the above mentioned problems.

Any model or simulation of a molecule can be of two different types of architectures. The first is the top-down containment structure. Starting from the top such as a membrane, you go down to different types of organelles, till the bottom to the functions they perform. In this way the entire architecture of the molecule is modelled. The other approach is the bottom-up approach. With the bottom up approach the dynamic reactions between molecules, and the rules and parameters that define these reactions are modelled. I will use the bottom up approach here, starting from the interactions between molecules I will move to a higher level where the global behaviour of an entire reaction is modelled. Paradigm will be used in modelling the behaviour of biological reactions, starting from interactions between the entities to global behaviours of these entities across various levels of descriptions and interactions. The bottom up approach is more suitable for modelling using Paradigm as the coordination language, since it can be applied in modelling the dynamic reactions between individual entities. Starting from the bottom, moving up to the top, the behaviour of the various components across various levels of description and interaction can be modelled using the bottom up approach. Therefore the bottom up approach is more suitable and has been applied in the models.

I will start with diagrammatic representations of very small entities such as a molecule and gradually add more detail. In this way I will show how it is possible to arrive at a complete biological reaction such as the binding of two molecules to form a bigger molecule. Based on this, the interactions of the molecules with other molecules in their environments and their global behaviours will be modelled. In the examples in this chapter I will present such a system whose behaviour can be specified in such a way. In the first example I will use an example of a molecule to explain the use of Paradigm in modelling biological processes, in the same way that it is applied in modelling software components. After that I will move on to a more complex reaction which is a biochemical reaction. In the biochemical reaction a protein acts as a catalyst in speeding up the entire reaction. The interactions between the protein and the other compounds will be modelled at an individual level of description and at a global level of behaviour.

There is no standard software system yet to visualise Paradigm notations therefore it was researched in the initial phase which software system is suitable for building models using Paradigm notations. GenMAPP and Rational Rose can be used to model the state transition diagrams. This was done in the initial phase to get familiar with these software systems so that it could be researched if they can be extended to dynamic software visualising Paradigm notations. I did not use these software systems later on to model the STD’s in this chapter; however it has been found out that it is possible to extend these software systems into dynamic systems. Although this project is to employ Paradigm’s methods to present the system behaviour specifications and coordination between relevant components, UML notions still can be of help for us to understand the overall concept of the system activities. This approach I believe will offer greater potential modelling flexibility and power. The software development community has been using concepts from UML to build complex systems, and that level of complexity and experience can be used in systems found in biology. UML is starting to be used to a limited extent within the biology community. Therefore two types of UML diagrams have been used; the collaboration diagram and the activity diagram. The activity diagram is quite similar to Paradigm’s concept of global behaviour diagram therefore it has been used to model the global behaviours of entities.

### 5.1.2 Process Description: Molecular adhesion

The first example which has been modelled is the binding of two molecules which is called molecular adhesion in biological terms. Molecular adhesion is the binding of a molecule to another molecule to form a bigger molecule. Molecular adhesion is regulated by specific adhesion molecules that interact with molecules on the opposing molecule or surface. The molecular binding can only take place if the molecules are in a particular range of each other. The molecule is first a static entity, only when it starts moving in search of another molecule to bind with, a dynamic process is initiated. This relationship of the molecule binding with another molecule will be explored in detail in the models to come. The relationship of the molecule binding with other molecules will illustrate the role of UML in modelling biological processes and how Paradigm's concepts can contribute to a better understanding.

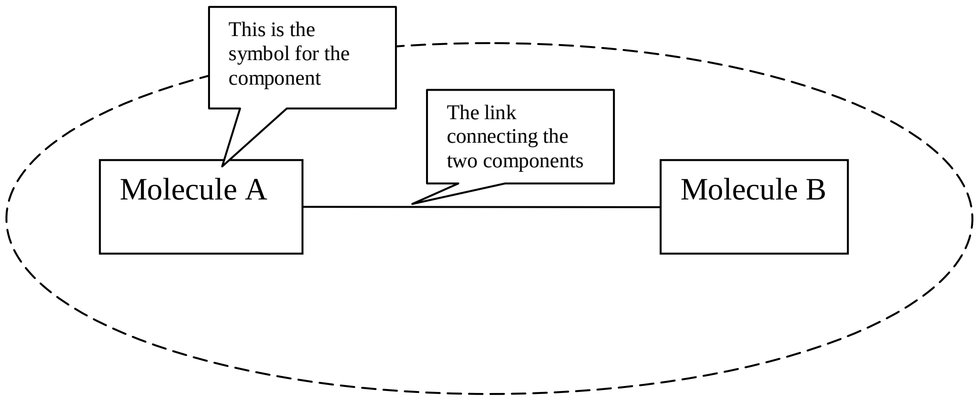

This is the symbol for the component
The link connecting the two components
Molecule A
Molecule B

**Figure 10, Collaboration diagram of two molecules collaborating with each other**

The collaboration between two molecules A and B is modelled in figure 10. Both molecule A and B are two components communicating with each other. When molecule A and molecule B are close enough the two molecules will bind or unite to form a single molecule and a weak bond will be established between the two. The molecules have to be in a certain range of each other for binding to take place.

### 5.1.3 STD and subprocesses

Now if I model this process of binding in more detail using a state transition diagram as in figure 11, all the states the molecules might pass through during binding would be clear.

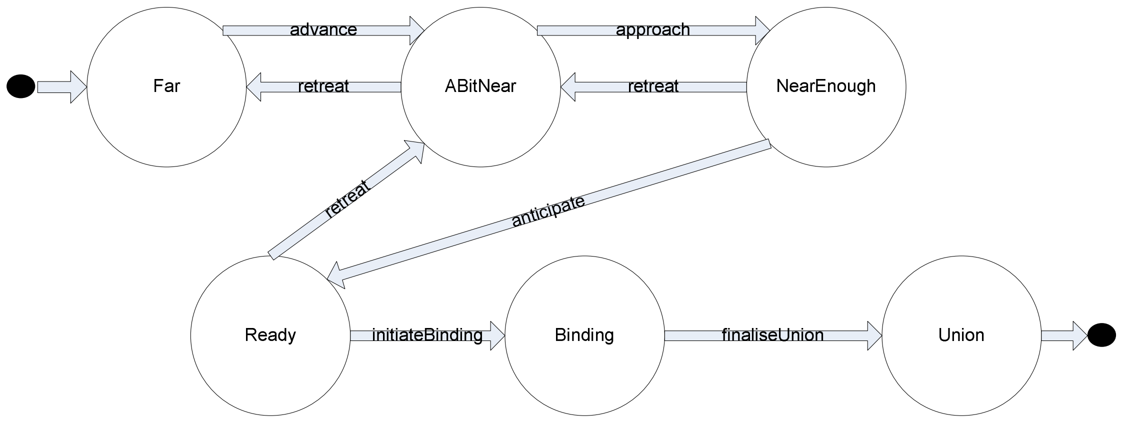


**Figure 11, STD Molecule**

**STD analysis:**

**Far:**

This is the initial state. The molecules are too far apart to bind with each other.

**ABitNear:**

In this state the molecules are a bit near towards each other, but still not in each other's range.

**NearEnough:**

The molecules are close in each other's range now and binding could be initiated.

**Ready:**

At this stage the molecules are ready to bind and are ready to initiate the binding process.

**Binding:**

The binding has been initiated and the molecules are binding with each other.

**Union:**

This is the final state, the binding has now been finalised and the molecules have united to form a bigger molecule.

**STD Description:**

In the first state the molecules are *Far* therefore nothing happens. When the molecules start moving towards each other, some kind of attraction takes place and the molecules now enter the state *ABitNear*. Here the molecules are advancing towards each other but are not within each other's range yet. The molecules keep on moving towards each other and once they are close enough they enter the state *NearEnough* where they become ready for binding and the binding could be initiated. At this stage the molecules move on to state *Ready* and the molecules start interacting between each other. This is the start of the binding process and the molecules now enter the state *Binding*. When the binding succeeds there is union of these two molecules and they unite to form a single molecule in the final state (*Union*).

This above mentioned scenario takes place if the binding succeeds. There are different stages from where the molecule retreats if something goes wrong in the binding process. From the state *ABitNear* the molecule can retreat to state *Far*. This happens when the molecule observes that the other molecule is not the right one for binding or if the binding did not succeed. Similarly from the state *NearEnough* the molecule retreats to state *ABitNear* if it had been waiting for too long for the other molecule to reach this state. From state *Ready* the molecule can retreat to state *ABitNear* if it again had been waiting for too long for the other molecule to get ready or if something else went wrong in the binding process.

This state transition diagram has six subprocesses representing six phases within the binding process. A subprocess temporarily restricts the complete behaviour; it is a (behavioural) part of the process. It can also be seen as a phase in the complete behaviour and during this phase the basic tasks to be performed are restricted to certain situations or subset of transitions. The subprocess explains the main process in more detail with the help of the different subprocesses that take place while the main process happens. The main subprocesses for figure 11 are shown in the following diagrams.

**Subprocesses**:

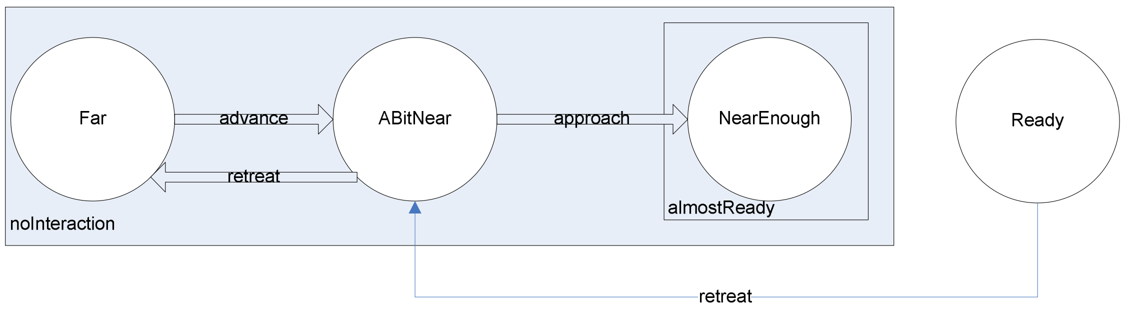


**Figure 12, NonInteracting**

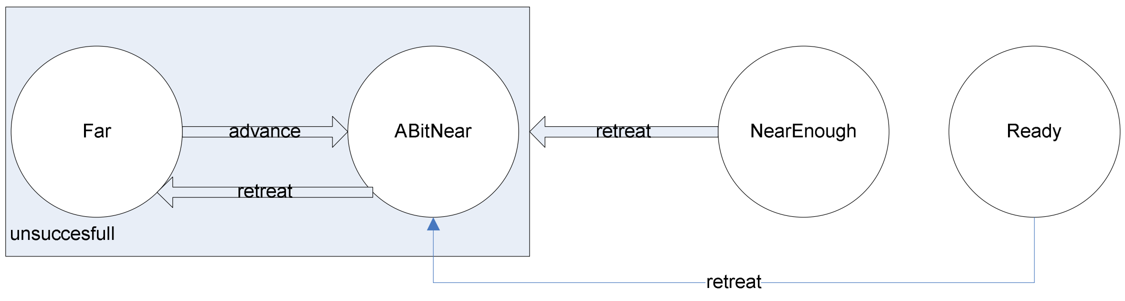


**Figure 13, Retreat**

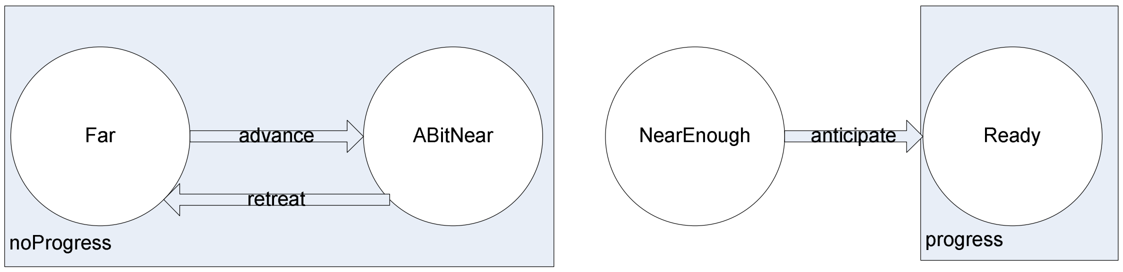


**Figure 14, Intercept**

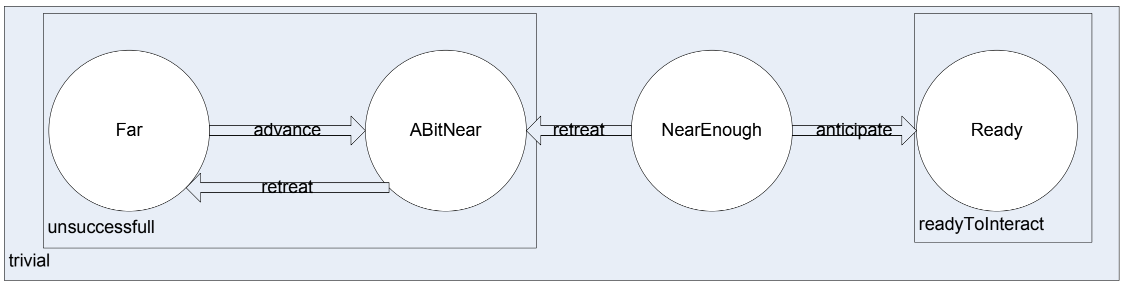


**Figure 15, TowardsInteraction**

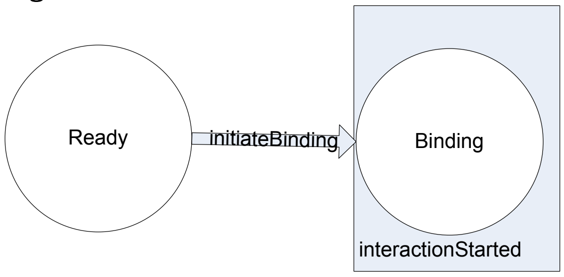


**Figure 16, Binding**

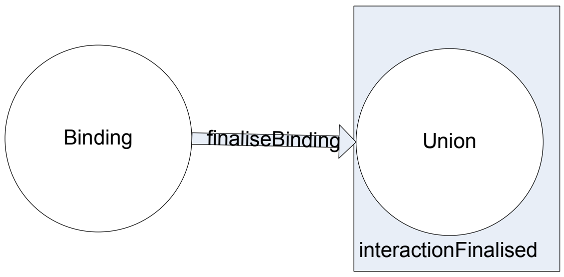


**Figure 17, Union**

The Molecule STD can be divided into one partition: *MoleculeBinding*, the subprocesses are shown as in figure 12-17. Six subprocesses are identified, *NonInteracting*, *Retreat*, *Intercept*, *TowardsInteraction*, *Binding*, and *Union* representing six phases of the molecule binding process. *NonInteracting* indicates currently the molecule has not started interacting with any other molecule; the molecule is preparing itself for the interaction. The molecule can proceed to prepare itself for the interaction and enter the trap *almostReady*, but it cannot enter the state *Ready* in this subprocess. From this subprocess the molecule can move on to the subprocess *TowardsInteraction* via the trap *almostReady* if it succeeds in preparing itself for the binding. If the molecule does not succeed in preparing itself for the binding there are two subprocesses for this situation; *Intercept* and *Retreat*. The molecule enters the subprocess *Intercept* via the trap *noInteraction* when it does not enter the required trap in time i.e. *almostReady* and is delaying the binding process. When the molecule is in the subprocess *Intercept* the molecule is interrupted from preceding any further, the state of the molecule is checked via this subprocess and the molecule can move on to the subprocess *TowardsInteraction* via the trap *progress* if it was in the required state (*NearEnough*), or remain in trap *noProgress* from where it will retreat to the subprocess *NonInteracting*. The molecule can also move from the subprocess *NonInteracting* directly to the subprocess *Retreat.* This happens if the molecule had entered the trap *almostReady* but the other molecule did not enter this trap. *Retreat* is the phase of unsuccessful interaction, the molecule retreats via this subprocess. When the molecule has entered the subprocess *TowardsInteraction* the molecule will have two choices either to retreat or proceed with the binding process choosing which by entering either trap *unsuccessfull* or *readyToInteract*. Subsequently these two traps will lead the molecule to the subprocess *NonInteracting* or *Binding.* In the subprocess *TowardsInteraction* if the molecule had entered the trap *readyToInteract* but has to retreat because the other molecule did not enter this trap it can do so via the trap *trivial*. In the subprocess *Binding* the molecule proceeds with finalising the binding process by entering the subprocess *Union*.

#### 5.1.3.1 Sequenced STD

To make the process explained in the previous section clear, in figure 18 I have modelled all these transitions of the molecules according to the different subproccesses that take place in sequence for the partition *MoleculeBinding*. The order of the subprocesses shown in figure 18 reflects the sequence in which the molecule will move from one subproccess to the other. The first subprocess *NonInteracting* is the one that takes place when the molecules are preparing themselves for the interaction. The molecule can retreat to the subprocess *NonInteracting* via the trap *noProgress* of the subprocess *Intercept*, trap *unsuccessfull* of the subprocess *Retreat* or the trap *unsuccessfull* of the subprocess *TowardsInteraction*. The molecule can retreat indirectly towards the subprocess *NonInteracting* when it first enters the subprocess *Retreat* via the trap *trivial* of subprocess *TowardsInteraction* and then retreat via the trap *unsuccessfull* of the subprocess *Retreat* to *NonInteracting*. The subprocess *Intercept* has been made to check the status of the molecule and to make it proceed towards the right direction. The third subprocess *Retreat* has been made for the unsuccessful finish of the subprocesses. If both the molecules succeed in the subprocess *NonInteracting* they move on to the subprocess *TowardsInteraction*, but if the molecules do not succeed in this subprocess, the subprocess *Retreat* or *Intercept* is entered. If the molecule is delaying the binding it will enter the subprocess *Intercept* via the trap *noInteraction* where its state will be checked. If it had entered the state *NearEnough* it will be made to move on to the the subprocess *TowardsInteraction* where it will directly enter the trap *readyToInteract,* else it retreats via the trap *noProgress* to the subprocess *NonInteracting*. The choice of making the molecule enter the trap *readyToInteract* directly from *progress* is to increase the chances of the successful completion of the binding process. For further increase in the probability of the successful completion of the entire binding process the other molecule which had entered trap *almostReady* could also be made to proceed via subprocess

*Intercept* to *TowardsInteraction*, where it will move directly to trap *readyToInteract* via trap *progress*. I did not choose to model it this way since I think this will force the molecule to go into a particular direction. The molecule will not have much freedom in its movement. This in my view makes the process less dynamic and restricts the process. The molecule can also move from the subprocess *NonInteracting* directly to the subprocess *Retreat*. This happens if the molecule had entered the trap *almostReady* but the other molecule did not enter this subprocess, from here the molecule retreats via the trap *unsuccessfull* to the subprocess *NonInteracting*. In the subprocess *TowardsInteraction* if the molecule enters the trap *unsuccessfull* it will retreat directly to the subprocess *NonInteracting* else it moves on to *Binding* via trap *readyToInteract*. In the subprocess *TowardsInteraction* if the molecule had entered the trap *readyToInteract* but has to retreat because the other molecule did not enter this trap it can do so via the trap *trivial* where it will enter the subprocess *Retreat*. From the subprocess *Retreat* the molecule will retreat to *NonInteracting* via trap *unsuccessfull*. In the subprocess *Binding* the molecule proceeds with finalising the binding process by entering the subprocess *Union*.

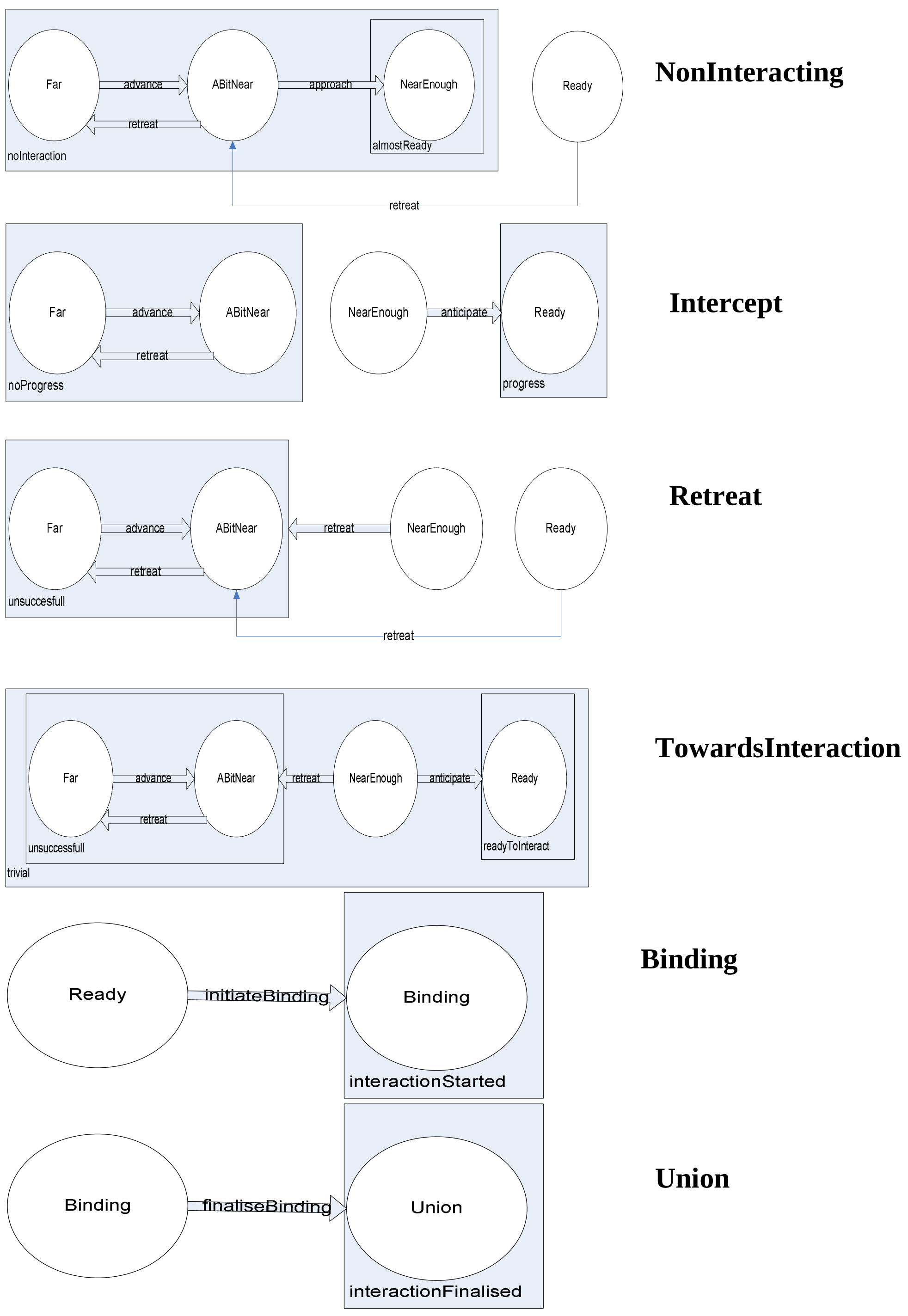


**Figure 18, Sequenced STD's**

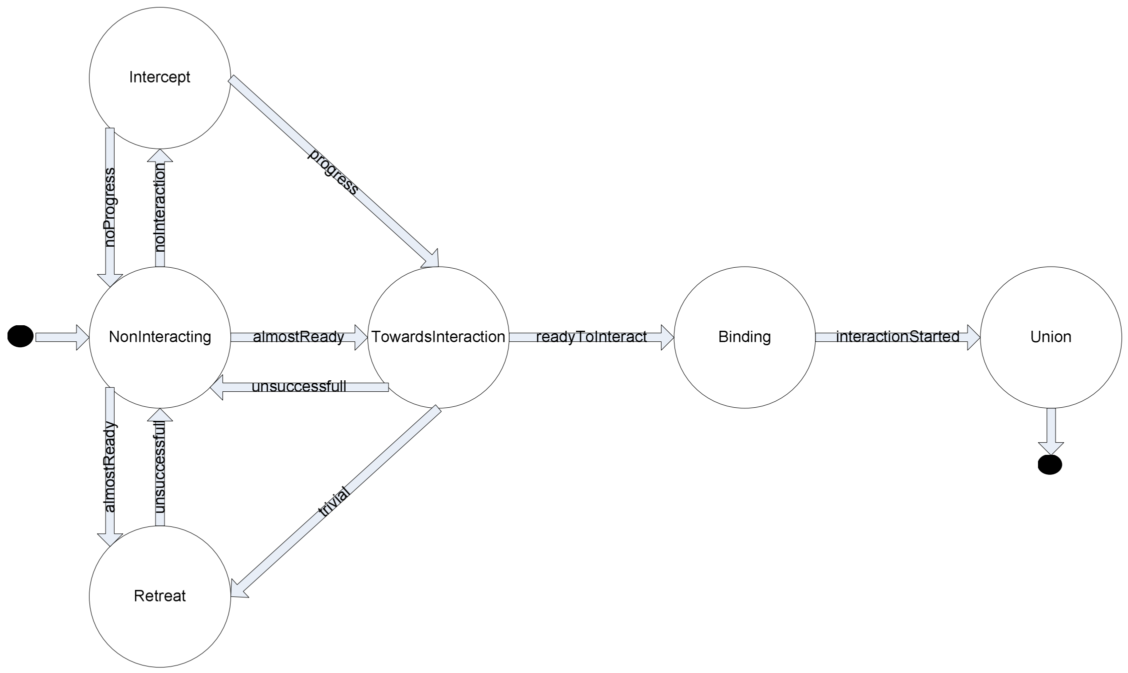
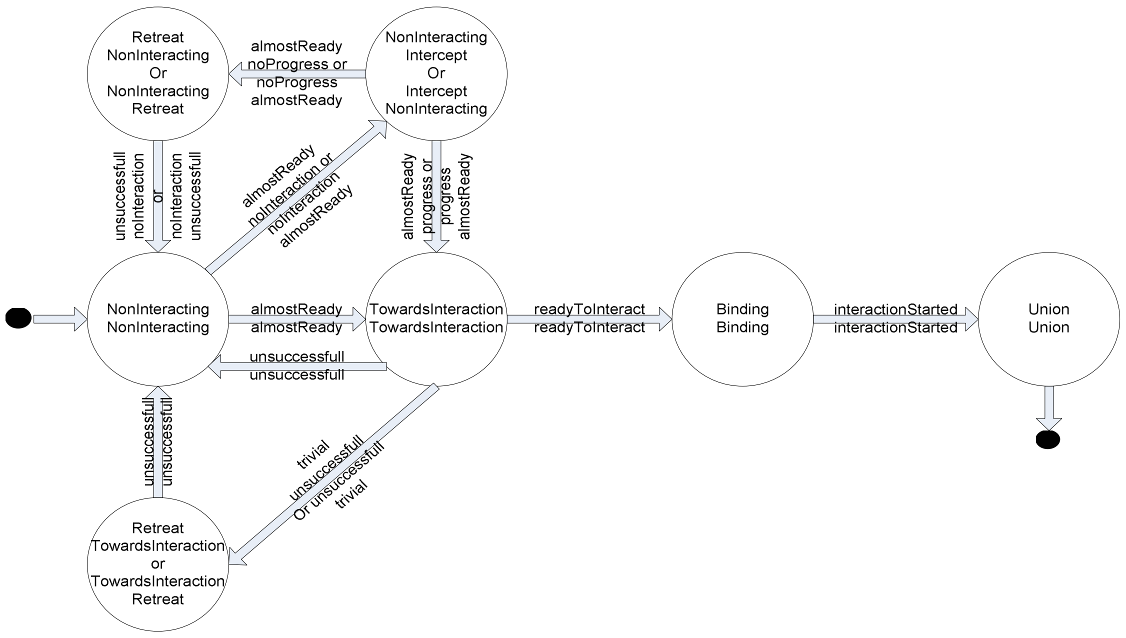


**Figure 19, The global process Molecule (MoleculeBinding) at the level of Partition MoleculeBinding**

The global behaviour of the Molecule STD at the level of partition *MoleculeBinding* for the process *'molecule binding'* is shown in figure 19. When the molecule is idle, no action is activated. Once the molecule starts moving it will advance towards the other molecule in *NonInteracting*, and approach the other molecule for interaction, in *TowardsInteraction.* Similarly as explained before all the phases (subprocesses) that the molecule can enter via the connecting traps are shown in the figure. The two dimensional view of the molecule binding process will make the entire process more clear as shown in figure 20. The figure shows all the possible combinations of the molecule with the other molecule via the different subprocesses and the connecting traps.



Retreat NonInteracting Or NonInteracting Retreat
NonInteracting Intercept Or Intercept NonInteracting
almostReady noProgress or noProgress almostReady
unsuccessfull noInteraction or noInteraction unsuccessfull
almostReady noInteraction or noInteraction almostReady
almostReady progress or progress almostReady
NonInteracting NonInteracting
almostReady almostReady
TowardsInteraction TowardsInteraction
unsuccessfull unsuccessfull
readyToInteract readyToInteract
Binding Binding
interactionStarted interactionStarted
Union Union
unsuccessfull unsuccessfull
trivial unsuccessfull Or unsuccessfull trivial
Retreat TowardsInteraction or TowardsInteraction Retreat

**Figure 20, Global behaviours for Molecule A and B at the level of Partition MoleculeBinding**

#### 5.1.3.2 Coordination Specification in Paradigm

Figure 10 modelled the collaboration between molecule A and molecule B. Now if I add a third entity into the collaboration such as in figure 21, whose role is only as a protocol which will make the collaboration taking place between the two molecules more clear and is not a physical entity. That entity I will call the Observer. The Observer is fulfilling the role of the manager, which will be to coordinate the tasks of its employees which are in this case molecule A and B. Generally speaking, Observer will act as an intermediator, managing all the molecules. The molecules will communicate via the Observer. This corresponds to the concept of Paradigm where a message is sent by the managing component, in this case the Observer with the help of the subprocesses and feedback is received via the connecting traps. In figure 22, the main choice is that the Observer is the overall 'manager' of all the molecules. Observer will monitor every progress and coordination taking place in the system. The Observer can mediate i molecules, meaning it can mediate more than one molecule at a time. Each molecule has a STD as specified in STD and subprocesses section respectively. As this process is to create a union of two molecules, which means that the unified molecule to be created does not exist at the beginning of this process, thus it is not in the managing relationship before the process starts.

The Observer is the one keeping a check on the overall behaviour of the molecules and allows the molecules to move from one subprocess to the other via the connecting traps. The Observer has only been added for modelling purposes. In reality there might not be such an entity that is keeping a check on the behaviour of the molecules, the molecules themselves might be doing this as shown in the previous section. The Observer is not influencing the behaviour but helps in showing how the overall behaviour of the molecules is adapting or changing according to the different states they are in, how the communication is taking place between the two molecules and how this affects their global behaviour. This corresponds to what I mentioned in chapter 3 about models that a model is used to 'model' the reality, it does not really explicitly do this, but implicitly helps in understanding the reality in a better way. Therefore the model that I propose here using the Observer might not be an explicit reflection of reality but it helps in understanding how that reality is taking place.

The idea of Paradigm is to look at the problems globally, when the manager process, in this case the Observer waits until an employee enters a trap: the time for the employee (molecule A and B) to perform certain functions without the manager knowing exactly the details how this employee process is executing the functions. The Observer just needs to know that the molecule has finished them, thus allowing the Observer to continue this coordinating task. The role of the different subprocesses introduced in the previous section will be clearer with the Observer. The Observer will make sure that the molecules are not waiting too long for each other to get ready to bind and that the right action is taken when this situation arises. It will prevent either molecule from delaying the binding process.

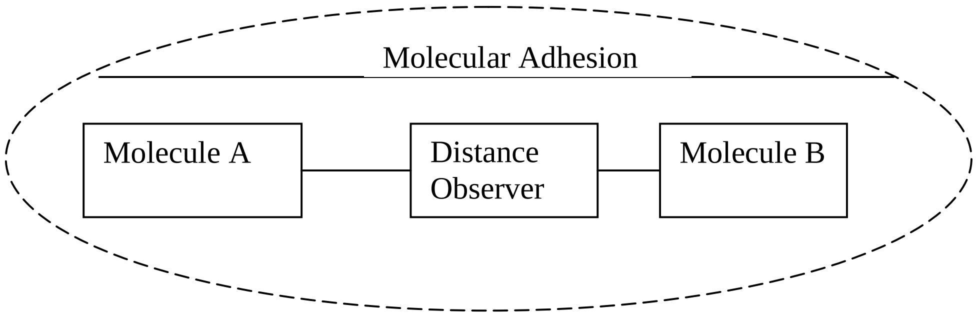


**Figure 21, Collaboration diagram 'Molecular Adhesion'**

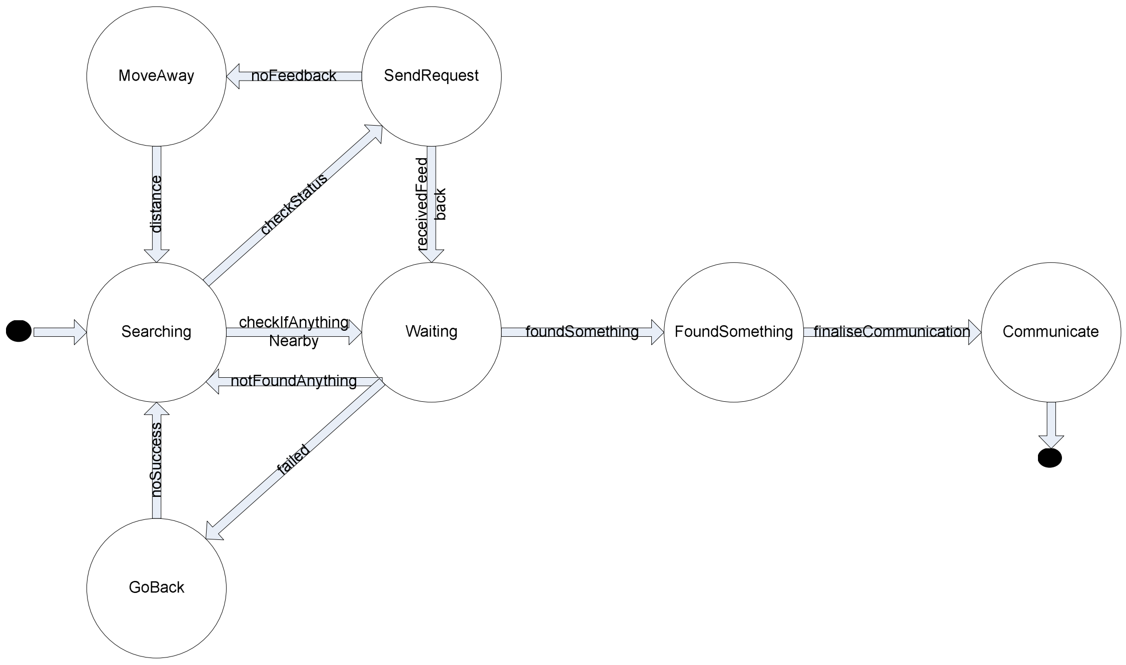


**Figure 22, STD Observer**

**STD analysis:**

**Searching:**

This is the first state of the Observer. The Observer in this state is searching for any molecules that can take part in the binding process.

**SendRequest:**

The Observer in this state has sent a request to the molecule to check its status.

**Waiting:**

The Observer is now waiting for the molecules to get close enough for binding to initiate.

**GoBack:**

The Observer retreats, the binding failed and the molecules go back to their initial state.

**MoveAway:**

The Observer moves back and searches somewhere else for molecules getting ready, the molecules retreat.

**FoundSomething:**

In this state the Observer has found two molecules that are ready to bind and have started the binding process.

**Communicate:**

This is the last state of the Observer, here the communication between the molecules that were binding gets finalised.

**STD Decription:**

In the STD Observer seven states have been identified, *Searching, SendRequest, Waiting GoBack, MoveAway, FoundSomething* and *Communicate*. With the help of this STD the communication and particularly the collaboration between the two different molecules will be observed or rather managed by the Observer. The first state of the Observer is *Searching*, where it searches for molecules that can take part in the binding process. When both molecules have entered the trap *almostReady* the Observer moves to state *Waiting* where it waits for both molecules to continue moving towards each other and get ready to bind where they enter the trap *readyToInteract*.

This happens if the first stage of the process of binding between the two molecules is completed successfully. If the first stage is not completed successfully there are different states from where the Observer retreats. The first one is if one molecule is in the required trap i.e. *almostReady* and the other molecule is not entering this trap the Observer will check the other molecule’s status and enter the state *SendRequest*. If the delaying molecule just had entered the state *NearEnough* the molecules will continue with the binding process. The Observer will continue and move to state *Waiting*. If the molecule that was interrupted had not entered the state *NearEnough* both the molecules will retreat and the Observer will move to state *MoveAway*. The Observer will move back to state *Searching* and continue searching somewhere else.

If everything did go well and both molecules are in trap *almostReady* or if via the subprocess *Intercept* the delaying molecule has also moved on and entered trap *readyToInteract* directly, the Observer moves to state *Waiting* where it will wait for both the molecules to get close enough and ready to bind. If both the molecules retreat directly the Observer moves back directly from state *Waiting* to state *Searching*. If one molecule enters the trap *readyToInteract* but the other keeps on delaying the binding process, after waiting for some time the Observer will send the feedback *notFoundAnything*. At this stage the molecules will retreat and the Observer moves to state *GoBack*. The Observer will move back to state *Searching* and it will continue searching somewhere else for molecules getting ready to bind. If both molecules enter the trap *readytoInteract* the Observer moves to state *FoundSomething*. From here on the molecules will finalise the communication between each other. The binding between the two molecules will now be initiated. The Observer moves to state *Communicate* and the two molecules finalise the binding process.

#### 5.1.3.3 Global Behaviour

In the section Sequenced STD's all the subprocesses were shown sequentially and the global behaviour of the molecules was shown for the partition *MoleculeBinding*. Figure 23 shows how the Observer will keep a check between molecule A and B and shows their global behaviours. As is clear from the figures both the molecules are influencing each other's behaviour and the state of one molecule determines the final state of the other molecule.

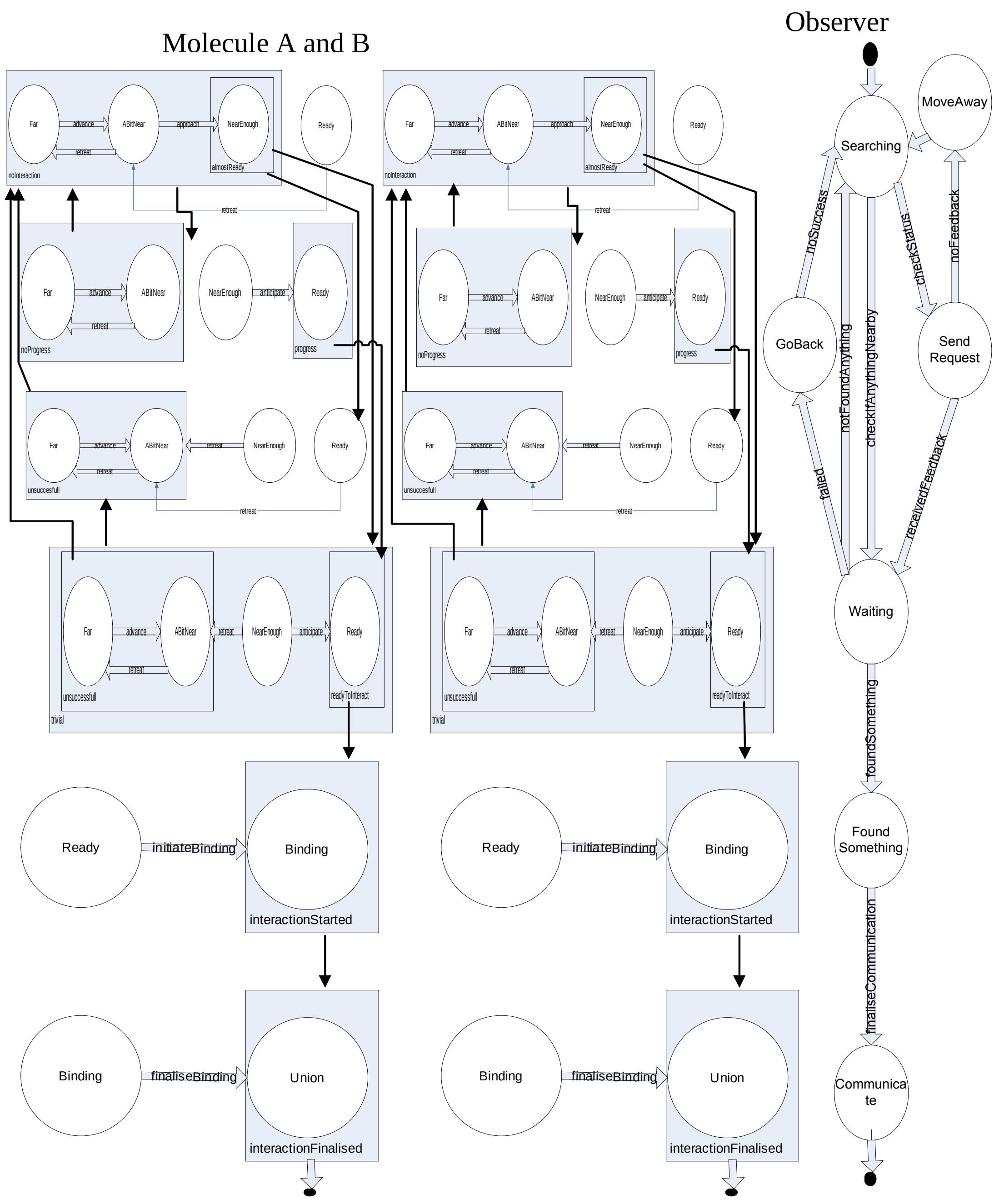


**Figure 23, Binding of Molecule A and B**

C1: Observer: Searching $\xrightarrow{\quad}$ SendReq

Molecule A: NonInteracting $\xrightarrow{\text{noInteraction}}$ Intercept

Molecule B: NonInteracting $\xrightarrow{\text{almostReady}}$ NonInteracting

C2: Observer: SendReq $\xrightarrow{\quad}$ Waiting

Molecule A: Intercept $\xrightarrow{\text{progress}}$ TowardsInteraction

Molecule B: NonInteracting $\xrightarrow{\text{almostReady}}$ TowardsInteraction

C3: Observer: Searching $\xrightarrow{\quad}$ SendReq

Molecule A: NonInteracting $\xrightarrow{\text{almostReady}}$ NonInteracting

Molecule B: NonInteracting $\xrightarrow{\text{noInteraction}}$ Intercept

C4: Observer: SendReq $\xrightarrow{\quad}$ Waiting

Molecule A: NonInteracting $\xrightarrow{\text{almostReady}}$ TowardsInteraction

Molecule B: Intercept $\xrightarrow{\text{progress}}$ TowardsInteraction

C5: Observer: SendReq $\xrightarrow{\quad}$ MoveAway

Molecule A: NonInteracting $\xrightarrow{\text{almostready}}$ Retreat

Molecule B: Intercept $\xrightarrow{\text{noProgress}}$ NonInteracting

C6: Observer: SendReq $\xrightarrow{\quad}$ MoverAway

Molecule A: Intercept $\xrightarrow{\text{noProgress}}$ NonInteracting

Molecule B: NonInteracting $\xrightarrow{\text{almostReady}}$ Retreat

C7: Observer: MoveAway $\xrightarrow{\quad}$ Searching

Molecule A: Retreat $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

Molecule B: NonInteracting $\xrightarrow{\text{noInteraction}}$ NonInteracting

C8: Observer: MoveAway $\xrightarrow{\quad}$ Searching

Molecule A: NonInteracting $\xrightarrow{\text{noInteracting}}$ NonInteracting

Molecule B: Retreat $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

C9: Observer: Searching $\xrightarrow{\quad}$ Waiting

Molecule A: NonInteracting $\xrightarrow{\text{almostReady}}$ TowardsInteraction

Molecule B: NonInteracting $\xrightarrow{\text{almostReady}}$ TowardsInteraction

C10: Observer: Waiting $\xrightarrow{\quad}$ FoundSom

Molecule A: TowardsInteraction $\xrightarrow{\text{readyToInteract}}$ Binding

Molecule **B**: TowardsInteraction $\xrightarrow{\text{readyToInteract}}$ Binding

C11: Observer: FoundSom $\xrightarrow{\quad}$ Com

Molecule A: Binding $\xrightarrow{\text{interactionStarted}}$ Union

Molecule B: Binding $\xrightarrow{\text{interactionStarted}}$ Union

C12: Observer: Waiting $\xrightarrow{\quad}$ Searching

Molecule A: TowardsInteraction $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

Molecule B: TowardsInteraction $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

C13: Observer: Waiting $\xrightarrow{}$ GoBack

Molecule A: TowardsInteraction $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

Molecule B: TowardsInteraction $\xrightarrow{\text{trivial}}$ Retreat

C14: Observer: Waiting $\xrightarrow{}$ GoBack

Molecule A: TowardsInteraction $\xrightarrow{\text{trivial}}$ Retreat

Molecule B: TowardsInteraction $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

C15: Observer: GoBack $\xrightarrow{}$ Searching

Molecule A: TowardsInteraction $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

Molecule B: Retreat $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

C16: Observer: GoBack $\xrightarrow{}$ Searching

Molecule A: Retreat $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

Molecule B: TowardsInteraction $\xrightarrow{\text{unsuccessfull}}$ NonInteracting

## Consistency Rules for figure 23

The consistency rules for figure 23 shows all the different states of molecule A together with the states of molecule B. These rules specify how the transitions and phase changes take place. Rule C1 is fired if molecule A is delaying the binding process and rule C3 if molecule B is delaying the binding process. If molecule A was ready it continues with the binding process via rule C2. Rule C11 is fired when the binding is getting finalised. Similarly all the other possible transitions and phase changes for molecule A and B are shown in the consistency rules.

In the end if everything goes well the binding succeeds, the two molecules unite to form a single molecule. This molecule will have its own behaviour different from the previous molecules within some other biological reaction at a higher level from the previous single molecules. In this way you can model the behaviour of a big system according to all the different levels. This could give a clear overview of how the behaviour of the individual entities like for e.g. molecules is affecting the entire global behaviour within some tissue and how these individual molecules are interacting within their environment.

The Observer checks if the molecule is ready. If one molecule is ready but the other is not, after waiting for some time the Observer returns the molecule to state *ABitNear*. It cannot wait forever for the other molecule to get ready as has been mentioned before. Therefore it is useful to introduce a time element within the Observer. In that way the entire process would have to be finished within a certain time limit. Now if I add a timer to the Observer as in figure 24, it will help in managing the entire process within the given time interval. This will simplify the process of binding that was shown in the previous figures. The role of the timer can be seen as of interrupting the process to make sure it does not go on forever and neither does it keep on waiting forever for the molecules to become ready. From the state *Searching* the Observer starts the *TimeOut*. After that the process has to stop before the timer is finished.

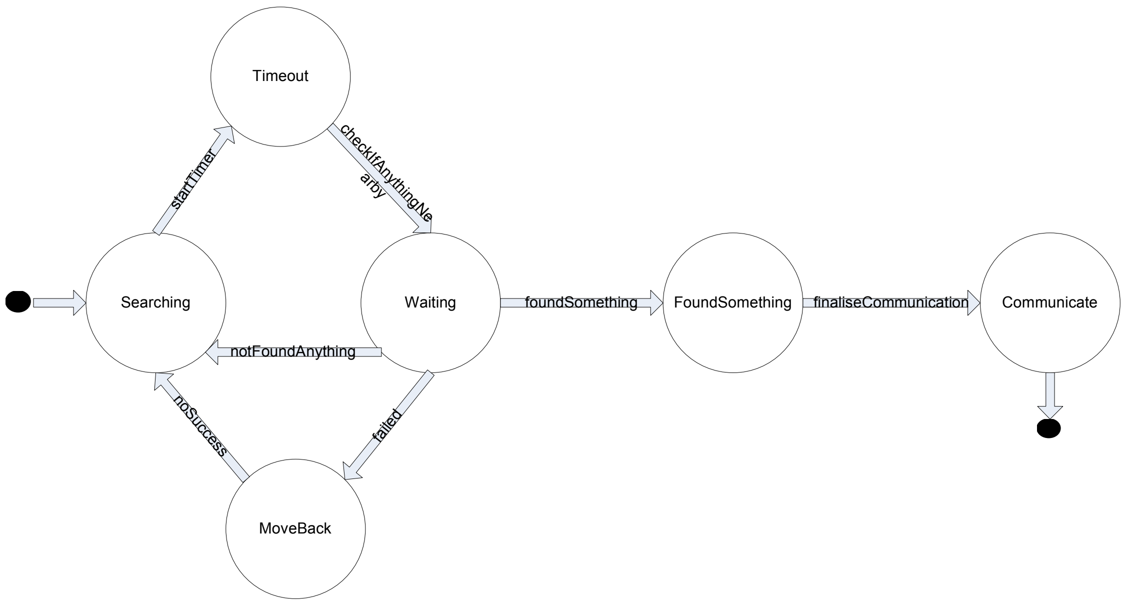


**Figure 24, STD Observer with timer**

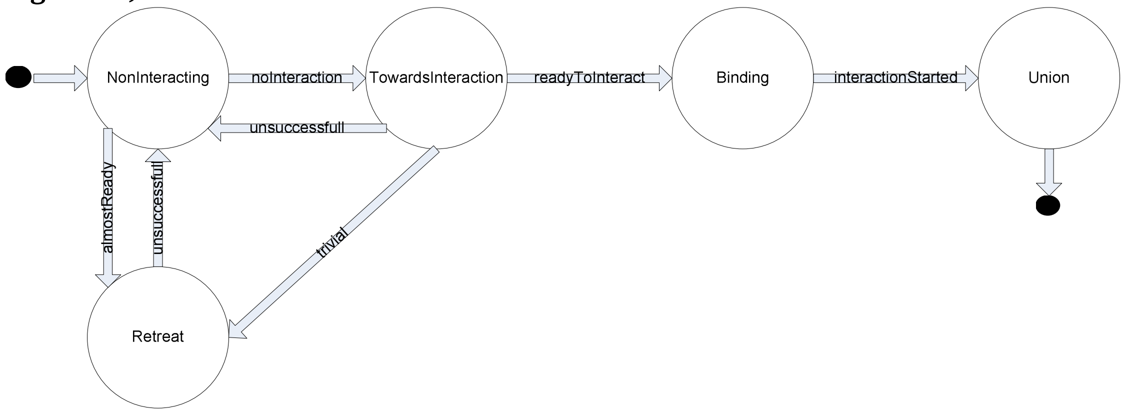
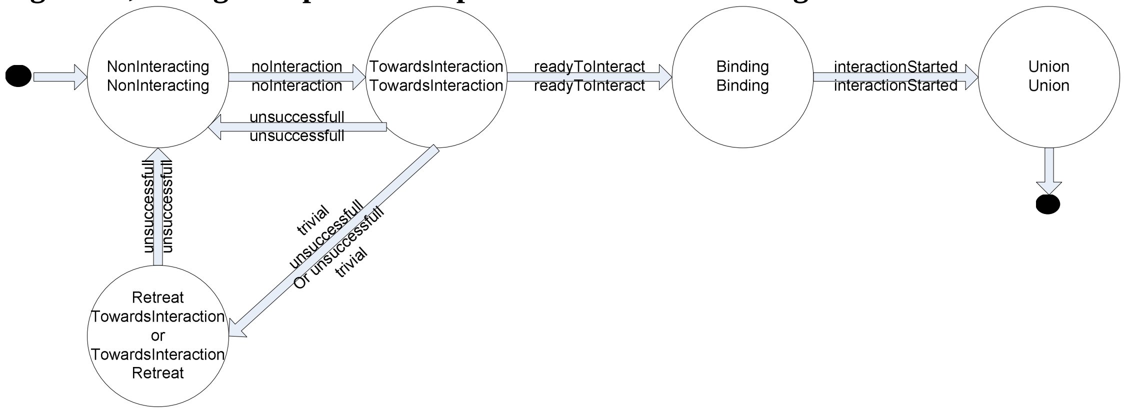


**Figure 25, STD global process for partition MoleculeBinding with Timer**



NonInteracting NonInteracting
noInteraction noInteraction
TowardsInteraction TowardsInteraction
readyToInteract readyToInteract
Binding Binding
interactionStarted interactionStarted
Union Union
unsuccessfull unsuccessfull
unsuccessfull unsuccessfull
trivial unsuccessfull Or unsuccessfull trivial
Retreat TowardsInteraction or TowardsInteraction Retreat

**Figure 26, Global behaviour of molecule A and B for partition MoleculeBinding with Timer**

The global process of the molecule with the timer would look like figure 25 and figure 26 shows the global behaviour for both molecule A and B. As is clear from the figures the role of the Observer has been simplified with the *TimeOut* state. In the previous figures if one molecule had reached the trap *almostReady* and the other molecule delayed the process, the Observer interrupted the delaying molecule with the subprocess *Intercept* and forced it to either retreat or move on. The subprocess *Intercept* is not needed now with the *TimeOut*. Now with the help of the *TimeOut* if both molecules do not reach the required traps in time the entire binding process terminates. If the molecules do not enter the other required traps in time before the timer finishes, the entire process will be stopped and the molecules will retreat via the connecting traps. The molecules move on from the subprocess *NonInteracting* to *TowardsInteraction* via the trap *noInteraction*. This is because the Observer has taken an observing role with the *TimeOut* state and does not observe each transition as carefully as before and lets the molecules move on from one subprocess to the other more freely. It is assumed now that in the second subprocess i.e. *TowardsInteraction* the decision to enter trap *readyToInteract* or *unsuccessfull* is taken in a short period of time. This example shows how a process can be modelled in different ways, giving different interpretations of the same process. The global behaviour of molecule A and B with the timer would be like the following figure.

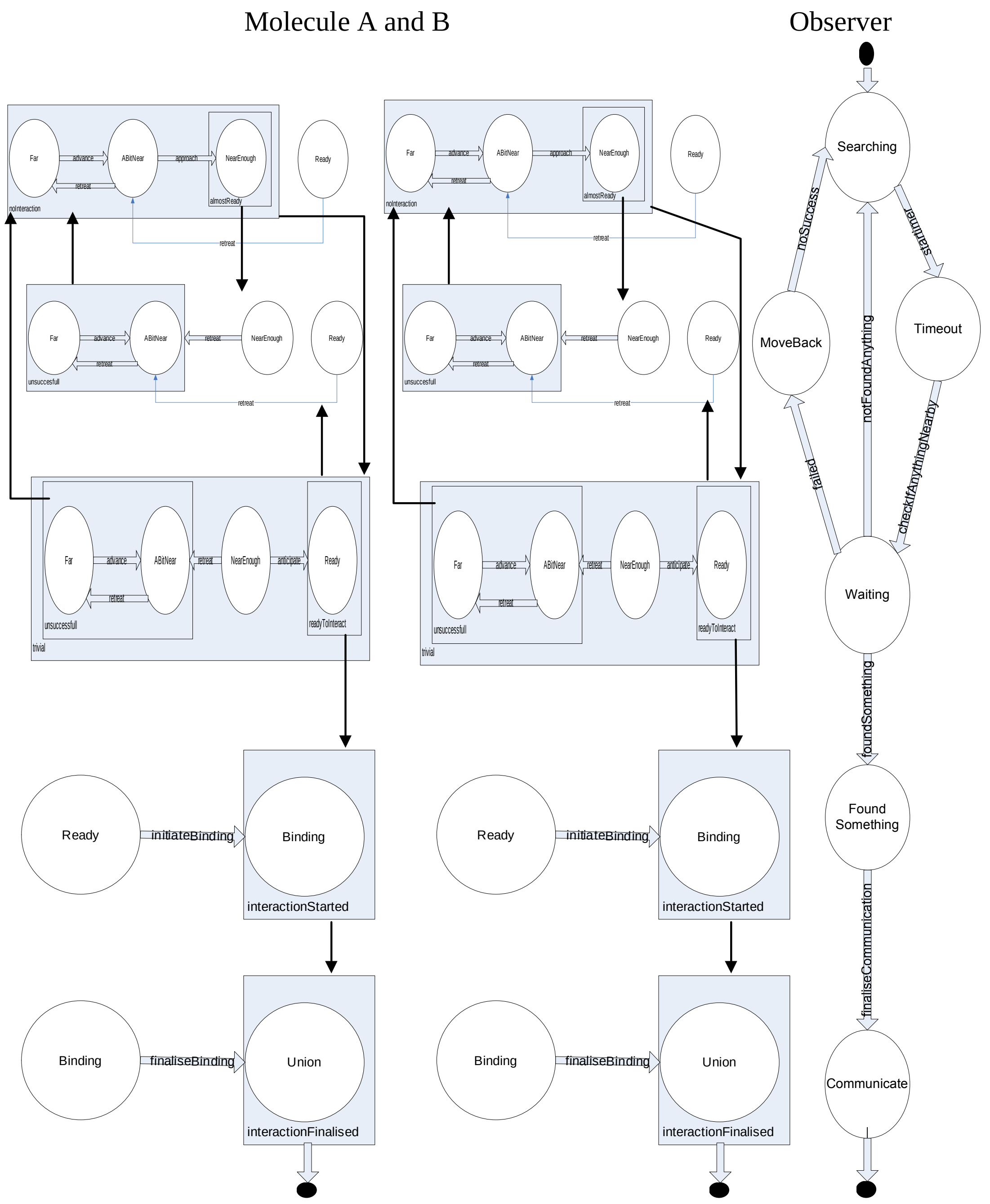


**Figure 27, Global behaviour of Molecule A and B with timer**

**Consistency Rules:**

The consistency rules start from *Searching* to *TimeOut* for the Observer. There are no subprocess changes for molecule A and B at this stage therefore no rules for them have been created. As the Observer moves from *TimeOut* to *Waiting* molecule A and B move from the subprocess *NonInteracting* to *TowardsInteraction*. The condition that should be fulfilled to fire this rule is that molecule A and B should move via the trap *noInteraction* from subprocess *NonInteracting* to subprocess *TowardsInteraction.* Rule C5 shows that Observer will move on when the molecule is ready to bind. The rest of the rules are delegated by the Observer in a similar fashion. The last rules C10 and C11 show the retreat of the molecules. The consistency rules for this reaction would look like the following:

| | | |
|---|---|---|
| C1: Observer: Searching | $\xrightarrow{\qquad}$ | Timeout |
| C2: Observer: TimeOut | $\xrightarrow{\qquad}$ | Waiting |
| Molecule A: NonInteracting | $\xrightarrow{\text{noInteraction}}$ | TowardsInteraction |
| Molecule B: NonInteracting | $\xrightarrow{\text{noInteraction}}$ | TowardsInteraction |
| C3: Observer: TimeOut | $\xrightarrow{\qquad}$ | Waiting |
| Molecule A: NonInteracting | $\xrightarrow{\qquad}$ | NonInteracting |
| Molecule B: NonInteracting | $\xrightarrow{\text{almostReady}}$ | Retreat |
| C4: Observer: TimeOut | $\xrightarrow{\qquad}$ | Waiting |
| Molecule A: NonInteracting | $\xrightarrow{\text{almostReady}}$ | Retreat |
| Molecule B: NonInteracting | $\xrightarrow{\qquad}$ | NonInteracting |
| C5: Observer: Waiting | $\xrightarrow{\qquad}$ | FoundSom |
| Molecule A: TowardsInteraction | $\xrightarrow{\text{readyToInteract}}$ | Binding |
| Molecule **B**: TowardsInteraction | $\xrightarrow{\text{readyToInteract}}$ | Binding |
| C6: Observer: FoundSom | $\xrightarrow{\qquad}$ | Com |
| Molecule A: Binding | $\xrightarrow{\text{interactionStarted}}$ | Union |
| Molecule B: Binding | $\xrightarrow{\text{interactionStarted}}$ | Union |
| C7: Observer: Waiting | $\xrightarrow{\qquad}$ | Searching |
| Molecule A: TowardsInteraction | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| Molecule B: TowardsInteraction | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| C8: Observer: Waiting | $\xrightarrow{\qquad}$ | MoveBack |
| Molecule A: TowardsInteraction | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| Molecule B: TowardsInteraction | $\xrightarrow{\text{trivial}}$ | Retreat |
| C9: Observer: Waiting | $\xrightarrow{\qquad}$ | MoveBack |
| Molecule A: TowardsInteraction | $\xrightarrow{\text{trivial}}$ | Retreat |
| Molecule B: TowardsInteraction | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| C10: Observer: MoveBack | $\xrightarrow{\qquad}$ | Searching |
| Molecule A: TowardsInteraction | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| Molecule B: Retreat | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| C11: Observer: MoveBack | $\xrightarrow{\qquad}$ | Searching |
| Molecule A: Retreat | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |
| Molecule B: TowardsInteraction | $\xrightarrow{\text{unsuccessfull}}$ | NonInteracting |

**Consistency Rules for figure 27**

## 5.2 Case study: 2.7.2.11

### 5.2.1 Process description

In the previous example I modelled the molecular binding process. In this second example I will model a more complex example which is of a biochemical reaction (fig 28) where a catalyst is involved in speeding up a biochemical reaction. I will apply the same concepts of Paradigm as I applied in the previous example to model this biochemical reaction.

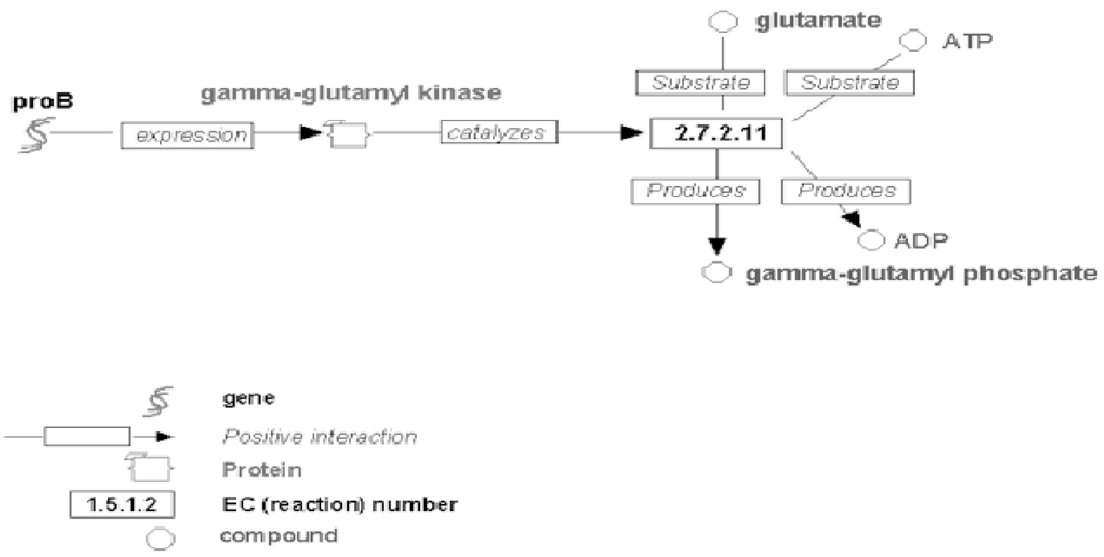


**Figure 28, Original Biological process [7]**

The reaction in figure 28 starts from the gene proB which takes part in the process Expression. The proB gene is transformed during the process Expression into the protein gamma-glutamyl kinase. This protein catalyzes the reaction 2.7.2.11. In the reaction 2.7.2.11 the substrates glutamate and ATP are transformed into gamma-glutamyl phosphate and ADP. The protein produced during Expression speeds up this reaction which acts as a catalyst (enzyme). The catalyst itself remains unchanged in the end and speeds up the reaction by lowering the activation energy of the compounds. Once the activation energy of the compounds is lowered they need less energy to react therefore the reaction is speeded up. What a catalyst is, what the number 2.7.2.11 denotes and what exactly gene expression is can be read in the chapter Basic Biological terms.

The unique quality about catalysts i.e. enzymes involved in reactions is that they do not produce any incomplete or side products. Therefore, during the modelling of this reaction I will assume that this entire biological process shown in figure 28 will be completed as is shown in the figure and no incomplete or side products will be produced. I will model this process in two stages. In the first stage the Expression of the proB gene can be modelled. The outcome of the process Expression i.e. protein gamma-glutamyl kinase, will act as a catalyst and will be involved in the reaction 2.7.2.11. In the second stage the reaction 2.7.2.11 will be modelled. Reaction 2.7.2.11 has three main compounds i.e. the catalyst, Glutamate and ATP reacting together to produce two compounds while the catalyst itself remains unchanged. The catalyst, Glutamate and ATP have their own unique behaviour; therefore they all will have their own STD.

To manage the process I will introduce the manager Cell. The Cell can be seen as managing the roles of the protein/catalyst and the reaction 2.7.2.11. The Cell is the manager of the protein/catalyst and the catalyst in turn is the manager of the reaction 2.7.2.11. Cell is the overall manager of all the proteins/catalysts and it can mediate i proteins/catalysts at a time. Each catalyst has a STD as is specified in the STD and subprocesses section to come. The Cell is not involved in the reaction 2.7.2.11 and assumes a mere observing role. The Cell lets the catalyst manage the entire reaction 2.7.2.11 and only when the catalyst has finished the reaction transforms the catalyst into the state protein. Cell is managing the role of the catalyst which exhibits two different behaviours. When the catalyst is not acting as a catalyst it's simply a protein but when it's involved in speeding up the reaction it gets the role of a catalyst which will have a different behaviour from when it's a protein. The Cell controls these transitions of the protein. I will explain this in more detail with the help of the STD and subprocesses diagrams. This process is to create ADP and gamma-glutamyl-phosphate, which do not exist at the beginning of this process, thus it is not in the managing relationship before the process starts.

### 5.2.2 STD and subprocesses

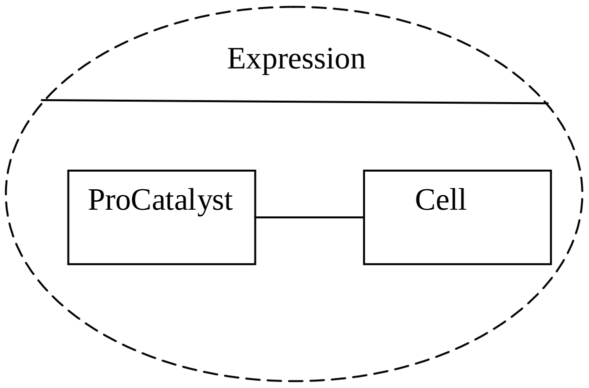


**Figure 29, Collaboration diagram of the ProCatalyst and Cell**

The Collaboration between the proB gene which is named as ProCatalyst and Cell is shown in figure 29. The name of this collaboration is Expression and the Cell is managing the process. The Cell and ProCatalyst are two components collaborating with each other. The STD's of the Cell and ProCatalyst will make their behaviour more clear.

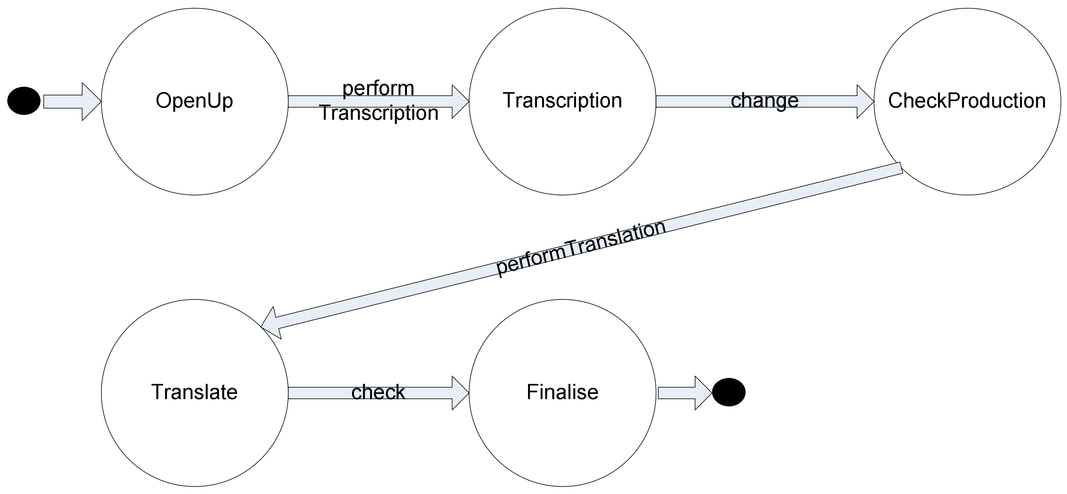


**Figure 30, STD Cell**

**STD analysis:**

**OpenUp:**

This is the first state of the cell where it opens up the gene.

**Transcription:**

The cell performs transcription on the protein.

**CheckProduction:**

The cell checks if the transcription has been performed well and there are no intermediate or incomplete products.

**Translate:**

If the right product has been produced the cell translates it.

**Finalise:**

The transformation of the protein is finalised and the protein is closed.

**STD description**

The STD of the Cell is shown in figure 30 where five stages of the cell have been identified. In the first stage the cell opens up the gene, this is done to make it ready for the transformations. In the second stage the cell performs transcription; the gene is than converted into mRNA. After transcription the intermediate product i.e. mRNA is checked if it has been transformed properly and is translated afterwards. Once translation is finished the gene is now a protein, therefore in the end the Cell finalises this transformation and closes the protein. All this is completed successfully and there are no intermediate or side products. As had been mentioned before we will assume that the process will be completed successfully, because if in the first stage something goes wrong the entire reaction cannot take place. We want to model the entire reaction therefore the first stage of this process should be completed successfully.

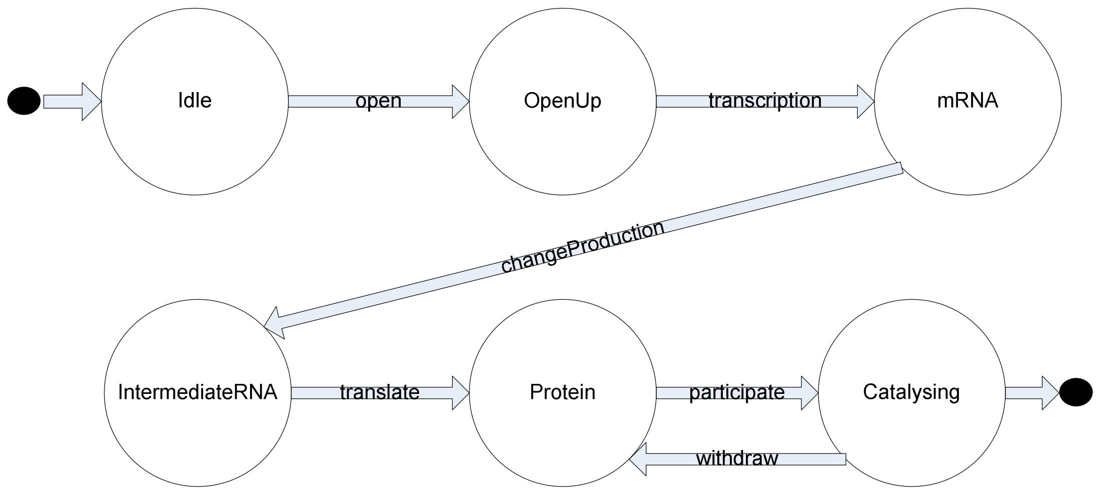


**Figure 31, STD ProCatalyst**

**STD analysis:**

**Idle:**
In the initial state the gene is idle.

**OpenUp:**
The gene is now open.

**mRNA:**
At this state the gene is getting transformed into mRNA.

**IntermediateRNA:**
The gene is now transformed into an intermediate mRNA.

**Protein:**
The intermediate mRNA is now transformed completely into a protein.

**Catalysing:**
This protein when involved in reactions acts as a catalyst.

**STD description**
The Cell modelled in figure 30 physically controls the transitions shown in figure 31 of the ProCatalyst. In the first state the gene is idle in state *Idle*. After the interference of the Cell it gets opened up and is prepared for transcription in state *Open*. Once the gene is opened, the Cell performs transcription on the gene and it gets transformed into *mRNA* which is further transformed into intermediate *mRNA* in state *IntermediateRNA*. This intermediate *RNA* is again changed and translated into a protein in state *Protein*. The Cell transforms the protein to state *Catalysing* when glutamate and ATP are near and the protein which is now a catalyst assumes the role of a manager. When it withdraws itself from the reaction the Cell transforms it back to the state *Protein* (ProCatalyst). When the protein is participating in the reaction 2.7.2.11 it assumes the state *Catalysing* (ActiveCatalyst) and when it withdraw it assumes the state *Protein*. The Cell controls

these transitions of the protein which will be explained further with the help of figure 42. The main subprocesses of the ProCatalyst would look like the following:

***Subprocesses***:

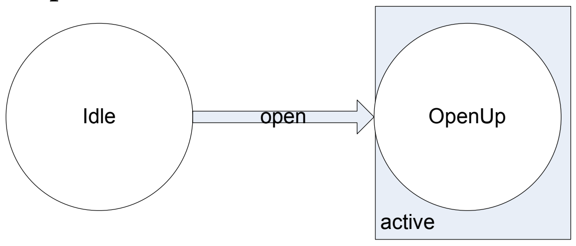


**Figure 32, MakeActive**

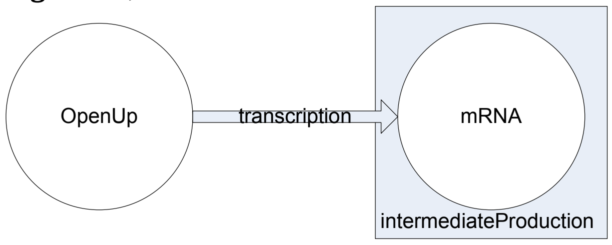


**Figure 33, PerFormTranscription**

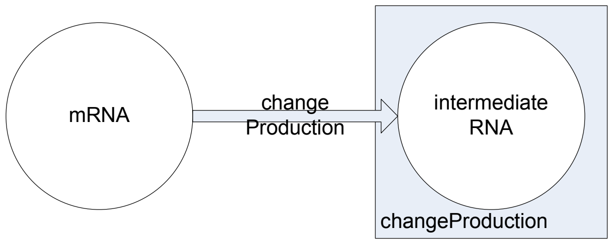


**Figure 34, Check**

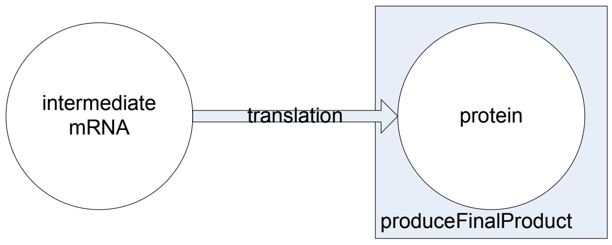


**Figure 35, PerformTranslation**

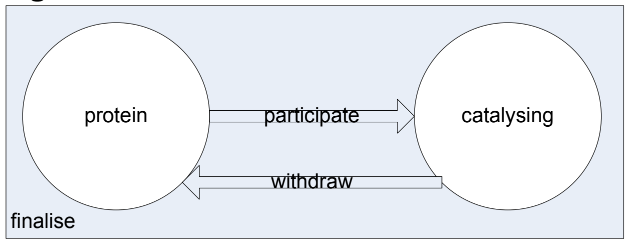
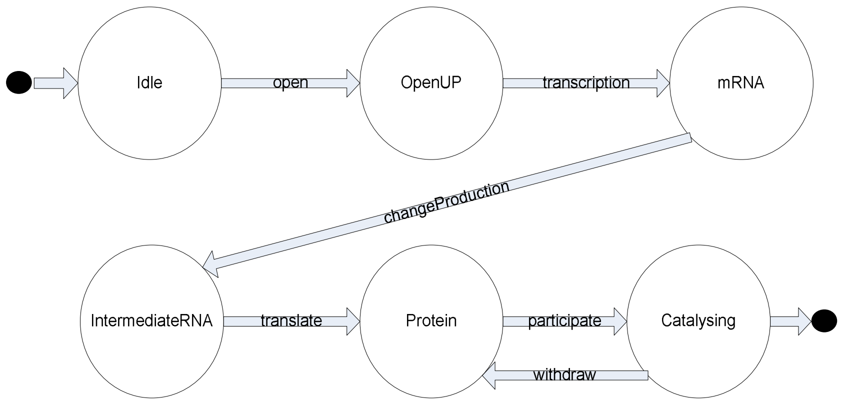


**Figure 36, FinaliseChanging**

The *ProCatalyst* STD has five subprocesses, as shown in figure 32-36 for the partition *ProduceProtein*. The subprocesses are; *MakeActive*, *PerformTranscription*, *Check*, *PerformTranslation,* and *FinaliseChanging* representing five phases of the gene transformation process. *MakeActive* indicates the gene is made active by the Cell. The Cell proceeds with transforming the gene in subprocess *PerformTranscription* and performs transcription on the gene. Here the gene enters the trap *intermediateProduct* and is transformed into intermediate mRNA. This intermediate mRNA is checked by the Cell in subprocess *Check* and made ready to be transformed further. In the final stages the intermediate mRNA is transformed completely into a protein in the subprocess *PerformTranslation* and it enters the trap *produceFinalProduct.* The transformed protein can assume two roles, one is of a protein when it's idle and not involved in any reactions and the other is of a catalyst when it's involved in reactions in the subprocess *FinaliseChanging*. This transformation is done in the final trap i.e. *finalise* where the protein can assume these two roles.

The global behaviour of the ProCatalyst is shown in the following figure.



**ProCatalyst:**

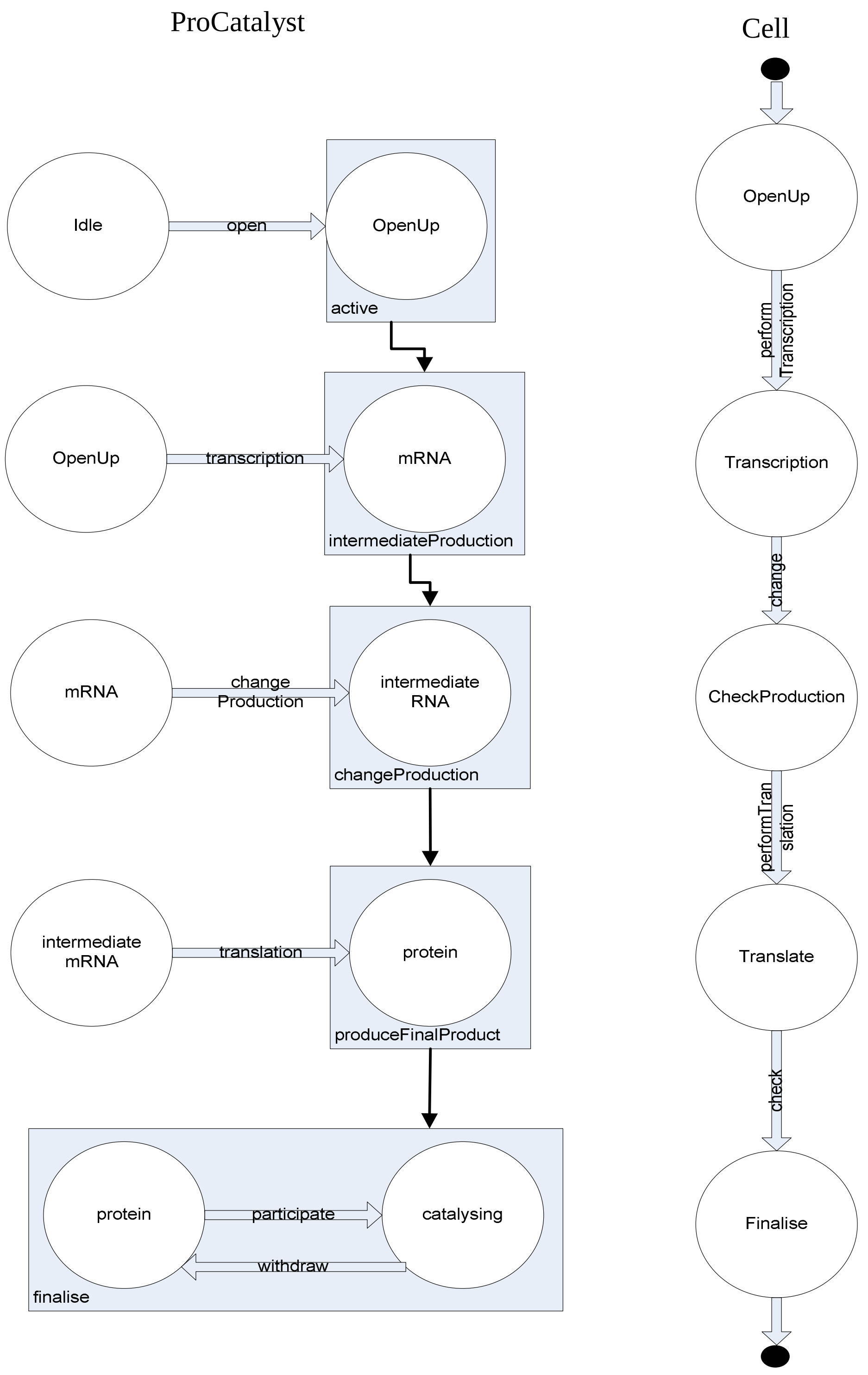


**Figure 37, Global Behaviour of the ProCatalyst**

C1: Cell: OpenUp $\xrightarrow{\qquad}$ Transcription

ProCatalyst: MakeActive $\xrightarrow{\text{active}}$ PerformTranscription

C2: Cell: Transcription $\xrightarrow{\qquad}$ CheckProduction

ProCatalyst : PerformTranscription $\xrightarrow{\text{intermedateProduction}}$ Check

C3: Cell: CheckProduction $\xrightarrow{\qquad}$ Translate

ProCatalyst : Check $\xrightarrow{\text{changeProduction}}$ PerformTranslation

C4: Cell: Translate $\xrightarrow{\qquad}$ Finalise

ProCatalyst : PerformTranslation $\xrightarrow{\text{produceFinalProduct}}$ FinaliseChanging

**Consistency Rules for figure 37**

In the previous models the Cell has been modelled as actively controlling the transformations of the gene. Another way of modelling the reactions would be where the Cell is just merely observing the transitions of the gene and not actively taking part in the overall transformation. The protein itself would be involved in all the transformations. The STD of the Cell when it has a mere observing role would be like the following figure:

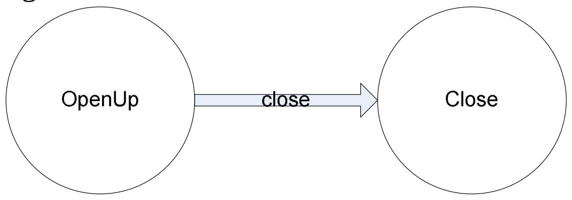


**Figure 38, STD Observing Cell**

The overall global behaviour of the ProCatalyst would look like the following, with the connecting traps representing the transitions:

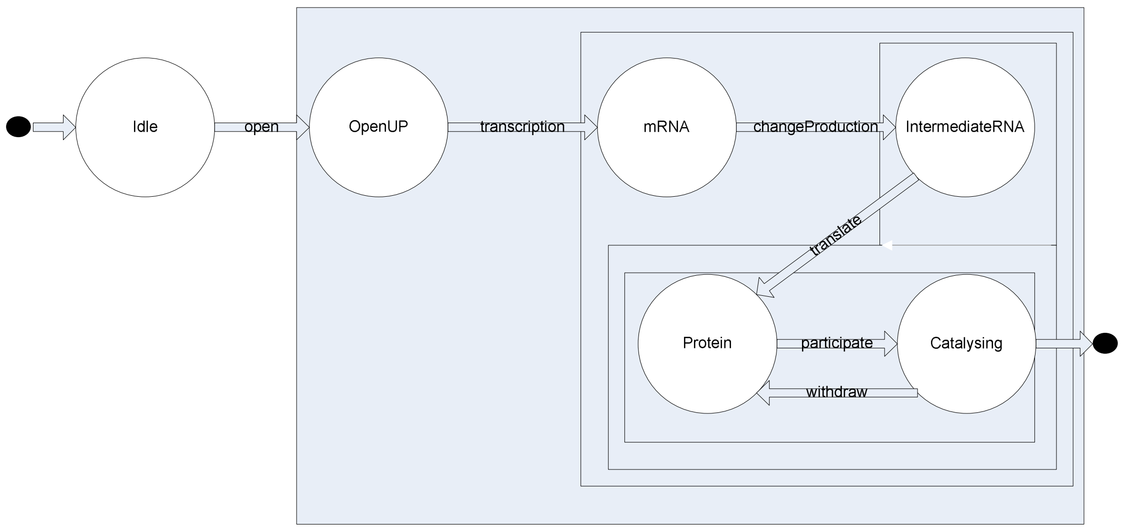


**Figure 39, Global Behaviour ProCatalyst**

The two different STD's of the Cell have been modelled to represent two different views of the same process. This would give the readers an idea how two models of the same process can give two different interpretations about their behaviour. This I hope will lead to better insights into the behaviour of a process.

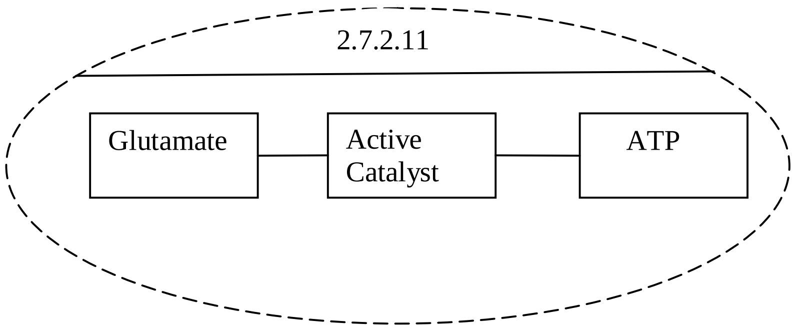


**Figure 40, Collaboration diagram of Glutamate and ATP**

The Collaboration between Glutamate and ATP is shown in figure 40. The number 2.7.2.11 denotes the name of the collaboration. The ActiveCatalyst which was first ProCatalyst is acting as the manager between the two since its now acting as a catalyst. The ActiveCatalyst is in fact ProCatalyst but since its now acting as a catalyst it has assumed the role of the manager and is actively taking part in the reaction.

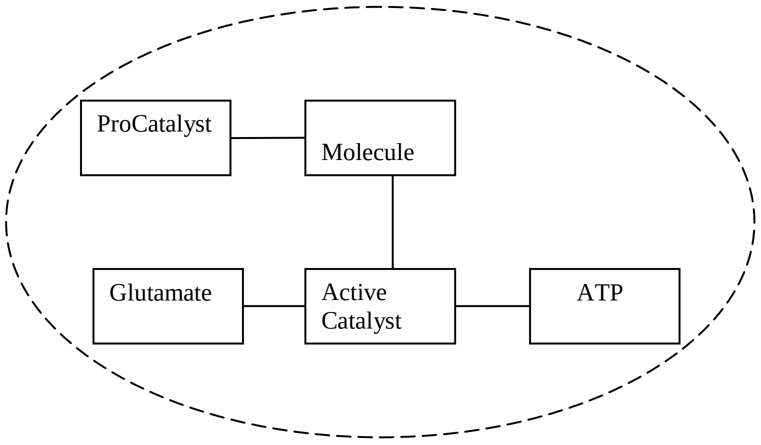


**Figure 41, collaboration diagram of the entire process**

The collaboration of the entire process is shown in figure 41. Cell is the manager of the ProCatalyst/ActiveCatalyst and the ActiveCatalyst is in turn the manager of Glutamate and ATP. ProCatalyst and ActiveCatalyst are two roles of a single compound i.e. the protein. The Cell will control these transitions of the protein from ProCatalyst to ActiveCatalyst. When the protein is ProCatalyst it is acting as a protein. Once it's transformed into ActiveCatalyst it will assume the role of the catalyst. When the compounds glutamate and ATP come closer to the protein the Cell will transform it from the state ProCatalyst to ActiveCatalyst and once the reaction is finalised the protein will assume the role of ProCatalyst again. These transitions are shown in figure 42 where the Cell is controlling the transitions of the protein.

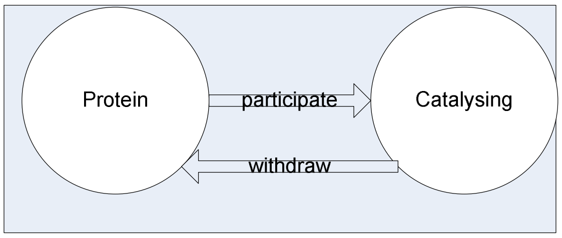


**Figure 42, STD ProCatalyst/ActiveCatalyst**

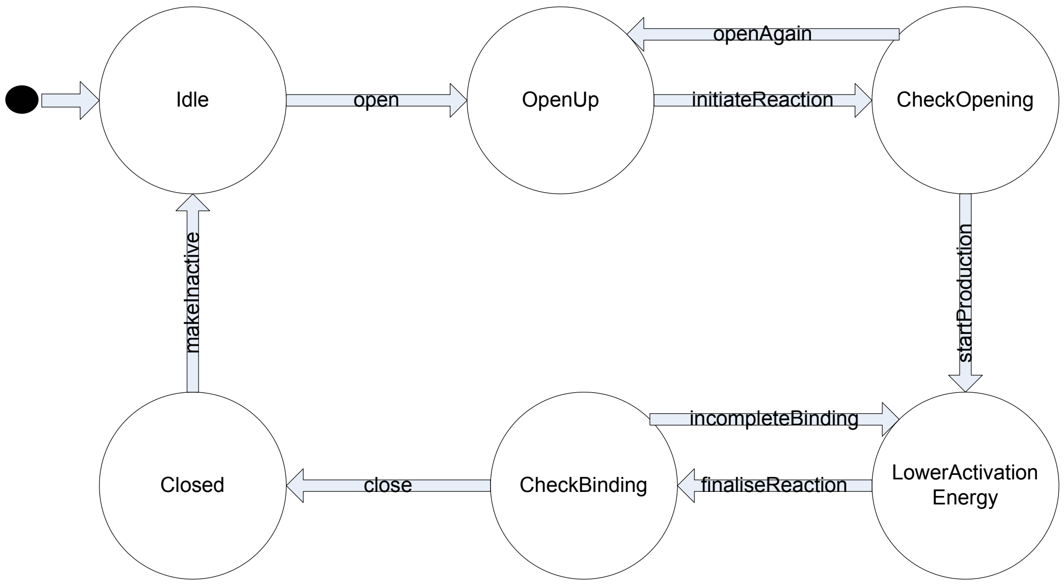


**Figure 43, STD ActiveCatalyst**

**<u>STD analysis:</u>**
**<u>Idle:</u>**
The protein is idle in the initial state and has not assumed the role of a catalyst yet.

**<u>OpenUp:</u>**
The protein has now assumed the role of the catalyst and is opening the substrates.

**<u>CheckOpening:</u>**
The catalyst checks if the right side of the compound had been opened.

**<u>LowerActivationEnergy:</u>**
The activation energy of the substrate is lowered by the catalyst.

**<u>CheckBinding:</u>**
The catalyst checks if the compounds have bonded properly.

**<u>Closed:</u>**
The catalyst closes the compounds and assumes the role of a protein again.

**STD description**

The STD of the ActiveCatalyst is shown in figure 43. The ActiveCatalyst has a separate STD from the ProCatalyst since the ActiveCatalyst will now be involved in catalysing the reaction and has assumed the role of the catalyst. The catalyst acts as a manager and is responsible for the production of ADP and gamma- glutamyl-phosphate from ATP and Glutamate in figure 44 and 45. The catalyst controls the reaction and the amount that is produced. The catalyst opens up the compound by performing the action *open* and entering the state *OpenUp*. Once the compound is open the catalyst checks if the right side has been opened by entering the state *CheckOpening*. If the right side of the compounds has been opened the catalyst lowers the activation energy of the compounds so that they can react with each other by moving on to the state *LowerActivationEnergy*. This is the energy required by the substrates i.e. the compounds to take part in the reaction. In this way the reaction is speeded up by the catalyst. If the right side has not been opened the catalyst moves back to state *OpenUp* and a new side of the compounds is opened. From the state *LowerActivationEnergy* the catalyst once again performs a check if the binding has been successful by moving to state *Checkbinding*. Here once again if the compounds do not bind properly the catalyst moves back to state *LowerActivationEnergy* and a new attempt is made to make the binding successful. The catalyst closes the final product produced during the reaction and itself enters the state *Idle* in the end.

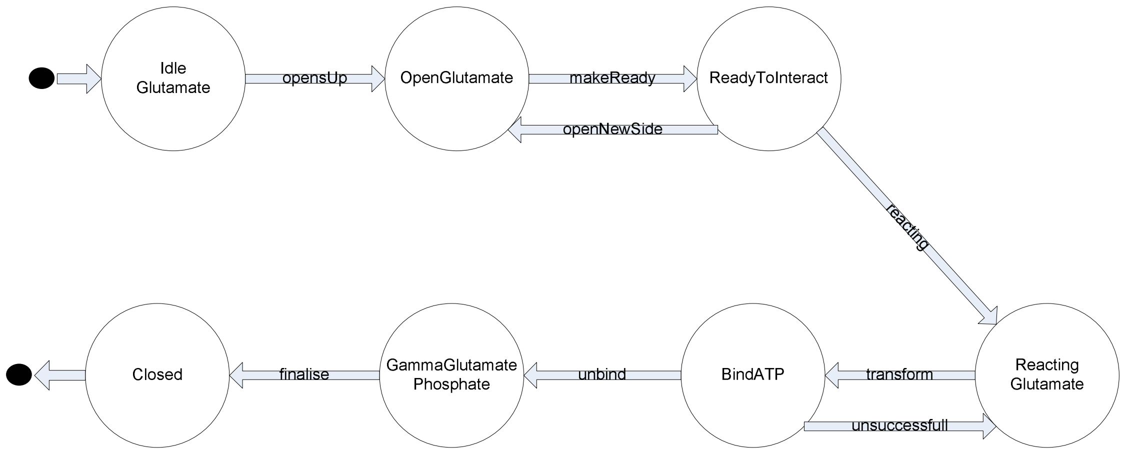


**Figure 44, STD Glutamate**

**STD analysis:**

**IdleGlutamate:**
In the initial stage before the reactions starts glutamate is idle.

**OpenGlutamate:**
In order for the reaction to take place glutamate is opened.

**ReadyToInteract:**
Glutamate becomes ready to participate in the reaction.

**ReactingGlutamate:**
Glutamate now reacts with the other compound.

**BindATP:**
Glutamate binds itself with ATP

**GammaGlutamatePhosphate:**
During the end of the reaction glutamate gets transformed into gamma-glutamate-phosphate.

**Closed:**
Once the reaction is finished the new compound i.e. gamma-glutamate-phosphate is closed.

**STD description**
Glutamate and ATP are first in an idle state with no action taking place. Once the catalyst is joined the compounds move from *Idle* to *Open* and get ready for the reaction. The reaction produces the two compounds in a controlled manner. Both Glutamate and ATP will therefore have their own STD, since they both change differently during the reaction.

Once glutamate is made *readyToInteract* by the catalyst it starts moving towards ATP. The catalyst checks if the right side has been opened at this stage, if not glutamate moves back and a new side is opened. Once the right side has been opened glutamate moves from *readyToInteract* to *reactingGlutamate* and starts reacting with ATP. Once it starts reacting with ATP it now binds itself with ATP and moves to state *BindATP*. Here again a check is performed by the catalyst to check if glutamate binds properly with ATP, if not it moves back to state *readyToInteract* and a new attempt is made for the binding. Once the binding is successful glutamate enters the final stages of the reaction and glutamate is transformed into gamma-glutamyl-phosphate. The catalyst unbinds itself from gamma-glutamyl-phosphate and closes the compound. The catalyst is involved in the entire reaction; therefore the entire process is completed successfully. If anything goes wrong the catalyst takes the right action and makes sure the compounds move in the right direction.

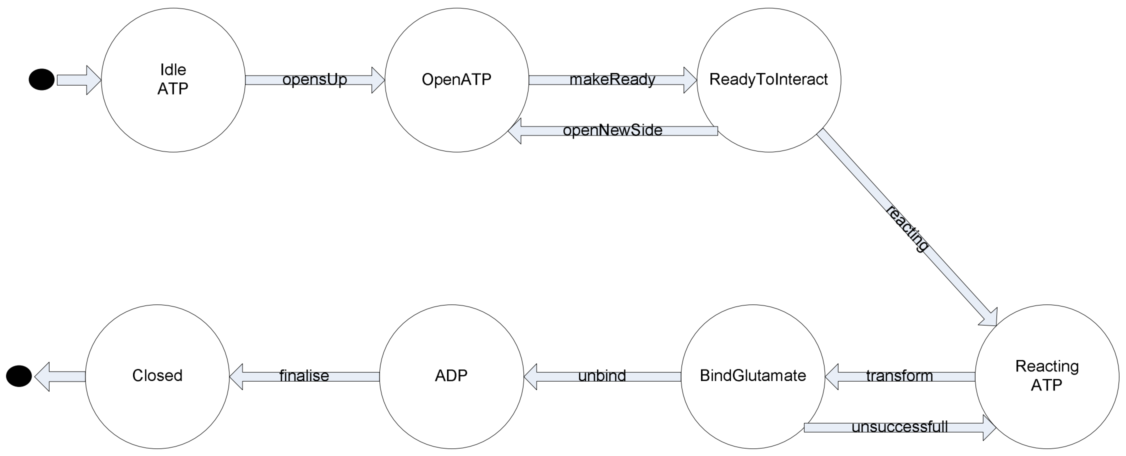


**Figure 45, STD ATP**

**STD analysis:**

**IdleATP:**
In the initial stage before the reaction starts ATP is idle.

**OpenATP:**
In order for the reaction to take place ATP is opened.

**ReadyToInteract:**
ATP prepares itself to participate in the reaction.

**ReactingATP:**
ATP now reacts with the other compound.

**BindGlutamate:**
ATP binds itself with glutamate.

**ADP:**
During the reaction ATP gets transformed into ADP.

**Closed:**
Once the reaction is finished the new compound i.e. ADP is closed.

**STD description**
ATP is transformed in the same way as glutamate by the ActiveCatalyst. Once ATP binds itself with glutamate it gets transformed into ADP and the catalyst closes it in the end. The subprocesses of glutamate and ATP would look like the following:

**Subprocesses: Glutamate**

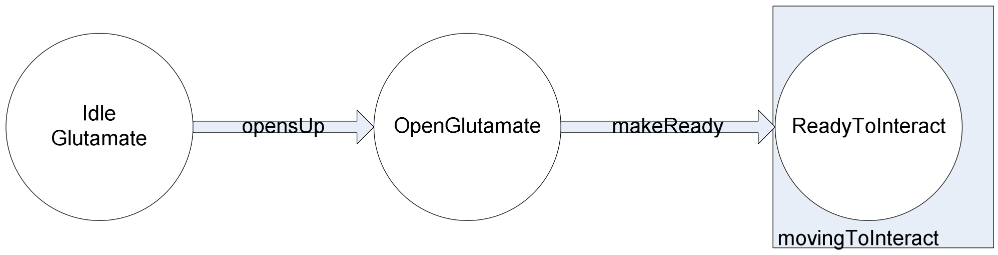


**Figure 46, ActiveGlutamate**

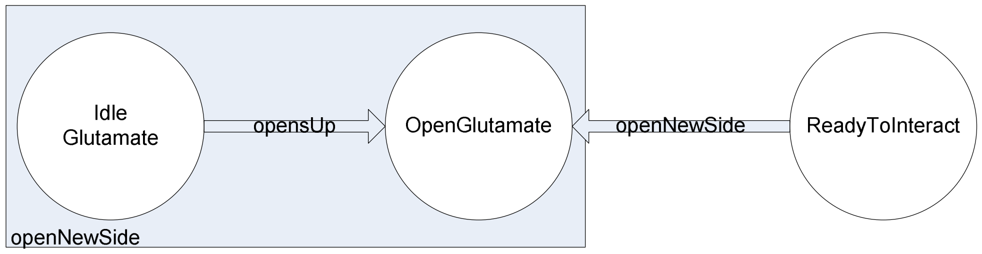


**Figure 47, OpenWrongSide**

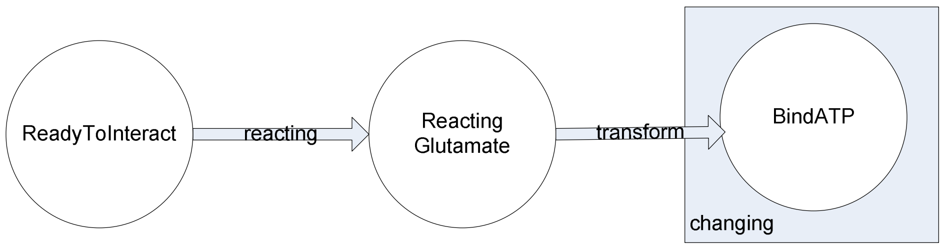


**Figure 48, Reacting**

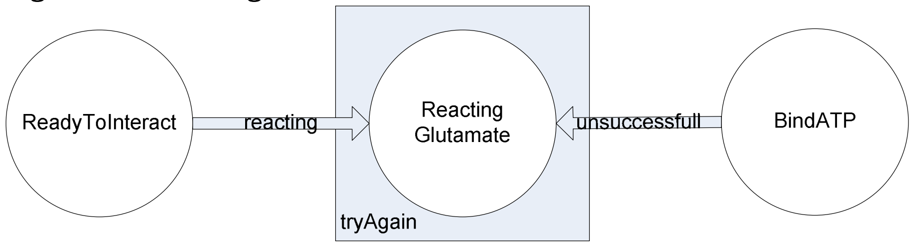


**Figure 49,  UnsuccessfullBinding**

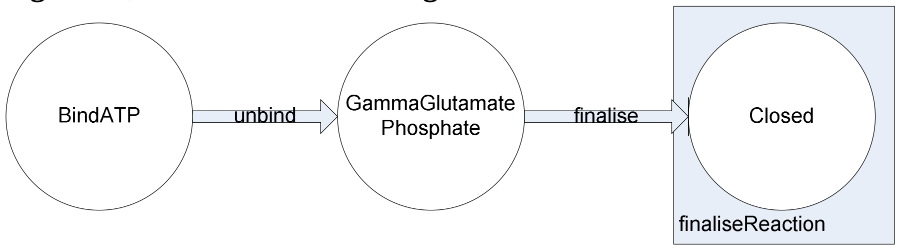


**Figure 50, ReactionComplete**

The subprocesses for glutamate in the partition '*produce GammaGlutamyl/ADP* ' are shown in figures 46-50 where five subprocesses are identified namely; *ActiveGlutamate*, *OpenWrongSide*, *Reacting*, *UnsuccessfullBinding* and *ReactionComplete*. *ActiveGlutamate* indicates that glutamate is becoming active for participating in the reaction; it can do this by entering the trap *active*.  If the wrong side is opened glutamate retreats from *ActiveGlutamate* to the subprocess *OpenWrongSide* from where it retreats to the state *OpenGlutamate* and another attempt can be made to open the right side. From *ActiveGlutamate* glutamate can proceed with preparing itself for the reaction by entering *Reacting*. From the subprocess *Reacting* glutamate can proceed with finalising the binding process and enter the subprocess *ReactionComplete*. If during binding something goes wrong like glutamate not being able to bind properly with the other compound it retreats from *Reacting* to *UnsuccessfullBinding*. In the subprocess *UnsuccessfullBinding* glutamate enters the trap *tryAgain* and retreats to state *ReactingGlutamate*. In the subprocess *ReactionComplete* Gamma-Glutamyl-Phosphate is produced from glutamate and the reaction is completed by entering the trap *finaliseReaction.*

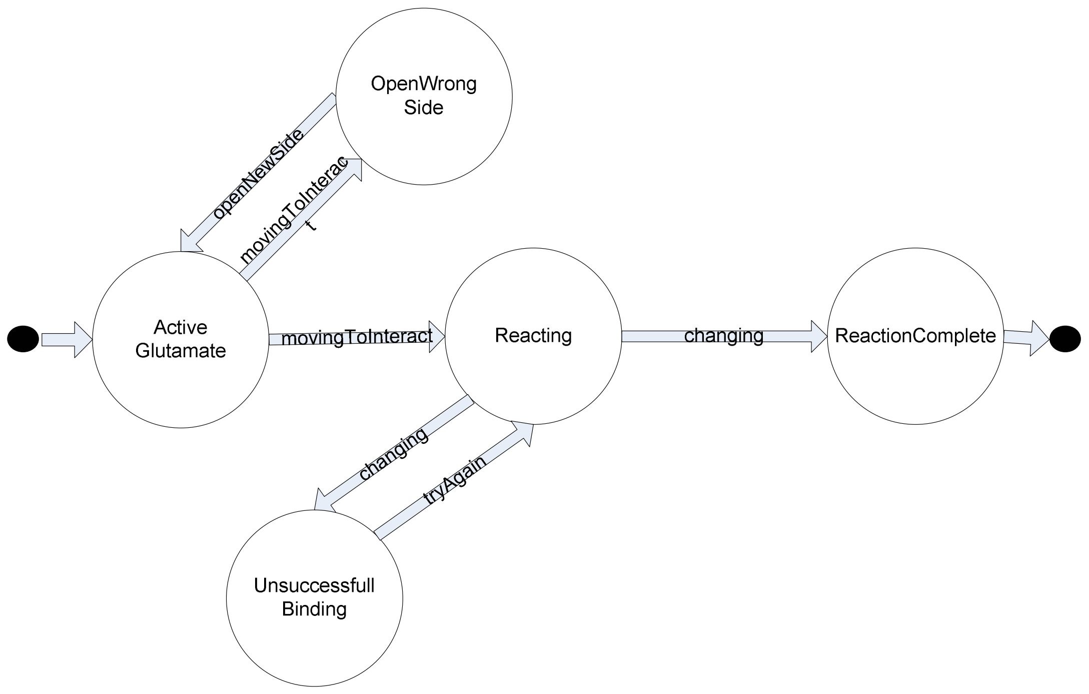


**Figure 51, Global process Glutamate for partition produce GammaGlutamyl/ADP**

The global behaviour of Glutamate STD at the level of partition '*produce GammaGlutamyl/ADP*' is shown in figure 51. When glutamate is idle, no action is activated. Once glutamate starts moving it will advance towards ATP in *ActiveGlutamate*, and approach ATP for interaction, in *Reacting.* Similarly as explained before all the phases (subprocesses) that glutamate can enter via the connecting traps are shown in the figure.

***Subprocesses: ATP***

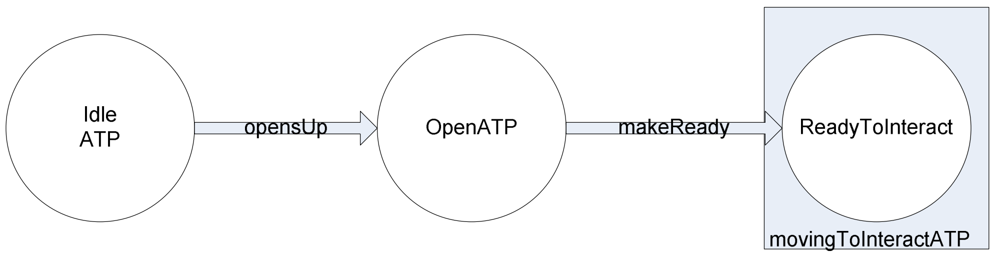


**Figure 52, ActiveATP**

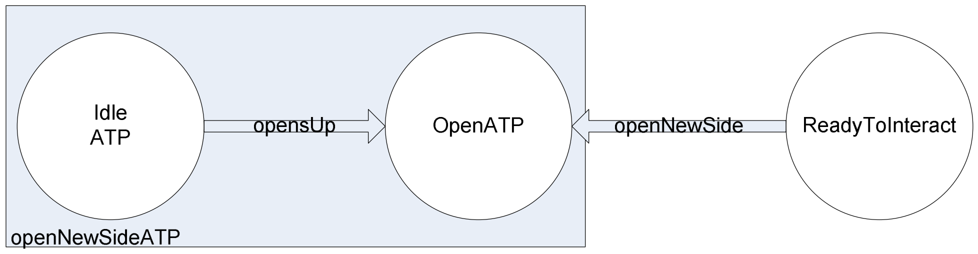


**Figure 53, OpenWrongSideATP**

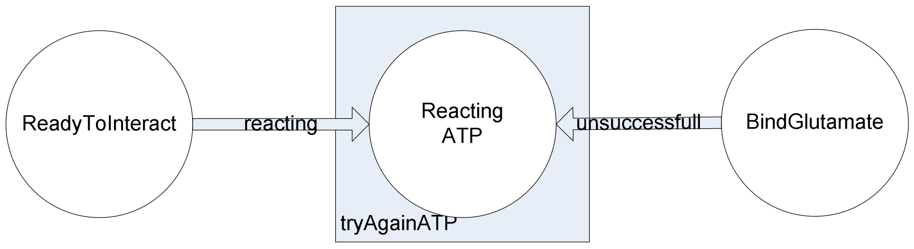


**Figure 54, UnsuccessfullBindingATP**

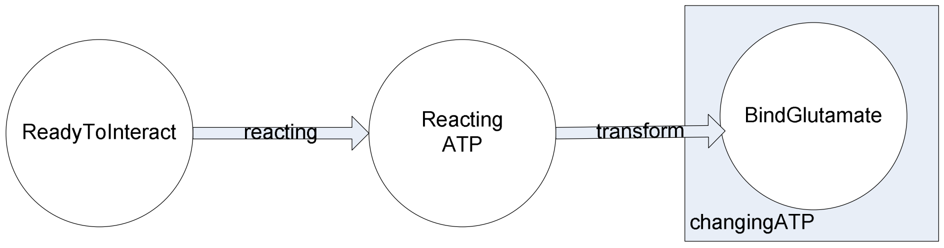


**Figure 55, ReactingATP**

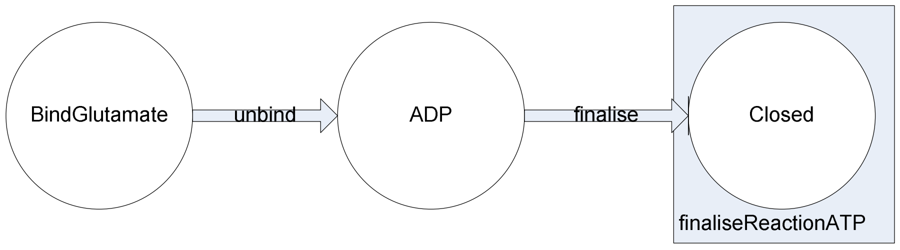


**Figure 56, ReactionCompleteATP**

The subprocesses for ATP are shown in figures 52-56 where five subprocesses are identified for the partition ‘produce GammaGlutamyl/ADP’ namely; *ActiveATP*, *OpenWrongSideATP*, *UnsuccessfullBindingATP*, *ReactingATP*, and *ReactionCompleteATP*. *ActiveATP* indicates that ATP is becoming active for participating in the reaction; it can do this by entering the trap *active*. If the wrong side is opened ATP retreats from *ActiveATP* to the subprocess *OpenWrongSideATP* from where it retreats to the state *OpenATP* and another attempt can be made to open the right side. From *ActiveATP* ATP can proceed with preparing itself for the reaction by entering *ReactingATP*. From the subprocess *ReactingATP* it can enter the final stages of the reaction by entering the subprocess *ReactionCompleteATP* where ADP is produced from ATP and the reaction is completed by entering the trap *finaliseReactionATP*. If during binding something goes wrong like ATP not being able to bind properly with the other compound it retreats from *ReactingATP* to *UnsuccessfullBindingATP*. In the subprocess *UnsuccessfullBindingATP* ATP enters the trap *tryAgain* and retreats to state *ReactingATP*.

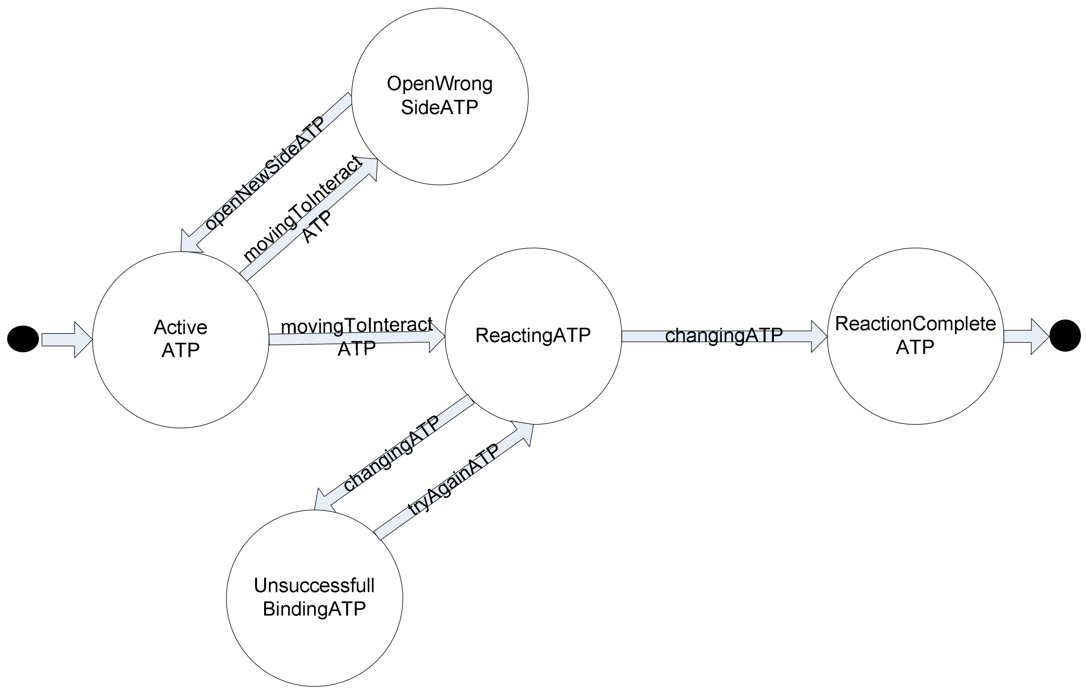


**Figure 57, Global process ATP for partition produce GammaGlutamyl/ADP**

The global behaviour of ATP STD at the level of partition '*produce GammaGlutamyl/ADP*' is shown in figure 57. ATP proceeds just like glutamate when it is idle, no action is activated. Once ATP starts moving it will advance towards glutamate in *ActiveATP*, and approach Glutamate for interaction, in *ReactingATP*. All the phases that ATP can enter via the connecting traps are shown in the figure.

### 5.2.3 Global behaviours

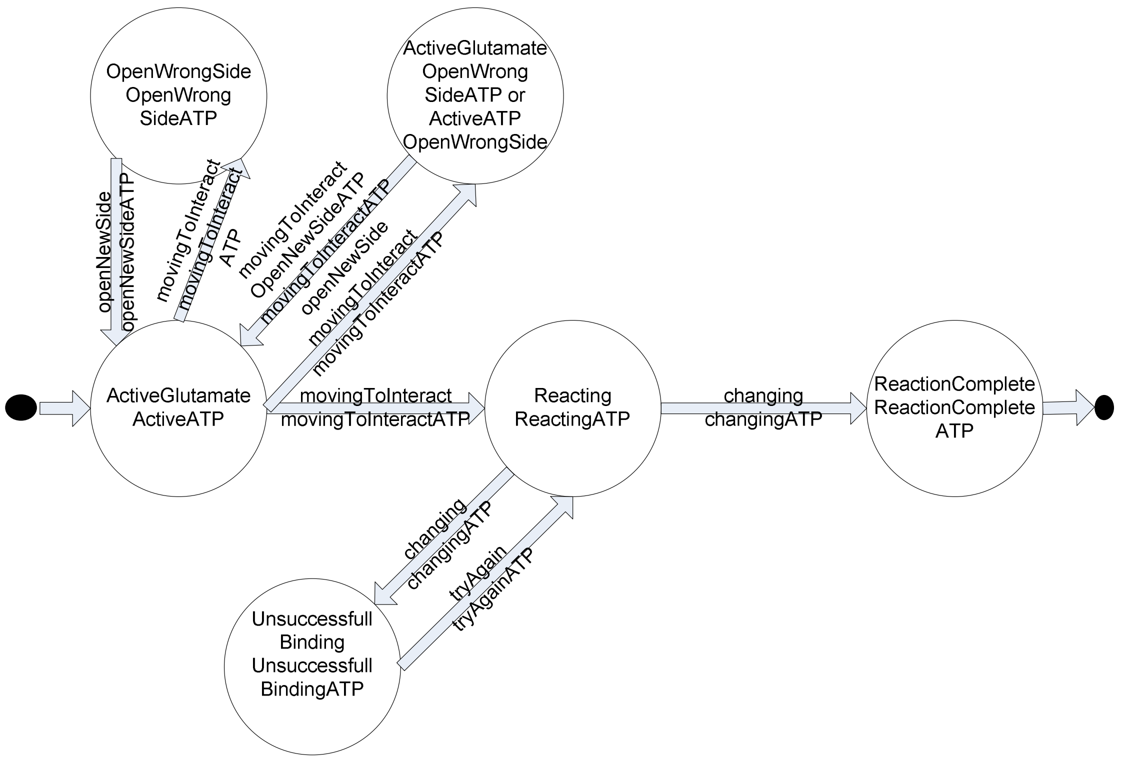


**Figure 58, Global behaviour Glutamate and ATP at the level of Partition produce GammaGlutamyl/ADP**

The two dimensional view of the partition produce GammaGlutamyl/ADP will make the entire process more clear as shown in figure 58. The figure shows all the possible combinations of glutamate with ATP via the different subprocesses and the connecting traps. If the wrong side is opened of ATP and glutamate both retreat back else if the wrong side of only ATP or Glutamate is opened one of them retreats and the other waits for it to proceed further. The catalyst makes sure the entire process is finished successfully therefore the compound waiting does not have to wait for too long for the other compound to proceed in the right direction. The compound continues waiting for the other compound and once the right side is opened both proceed in the right direction. During binding if one compound fails to bind properly the other also automatically fails to bind properly therefore both will retreat back and make another attempt to complete the binding process. The catalyst at this point takes action and makes sure that the binding does succeed.

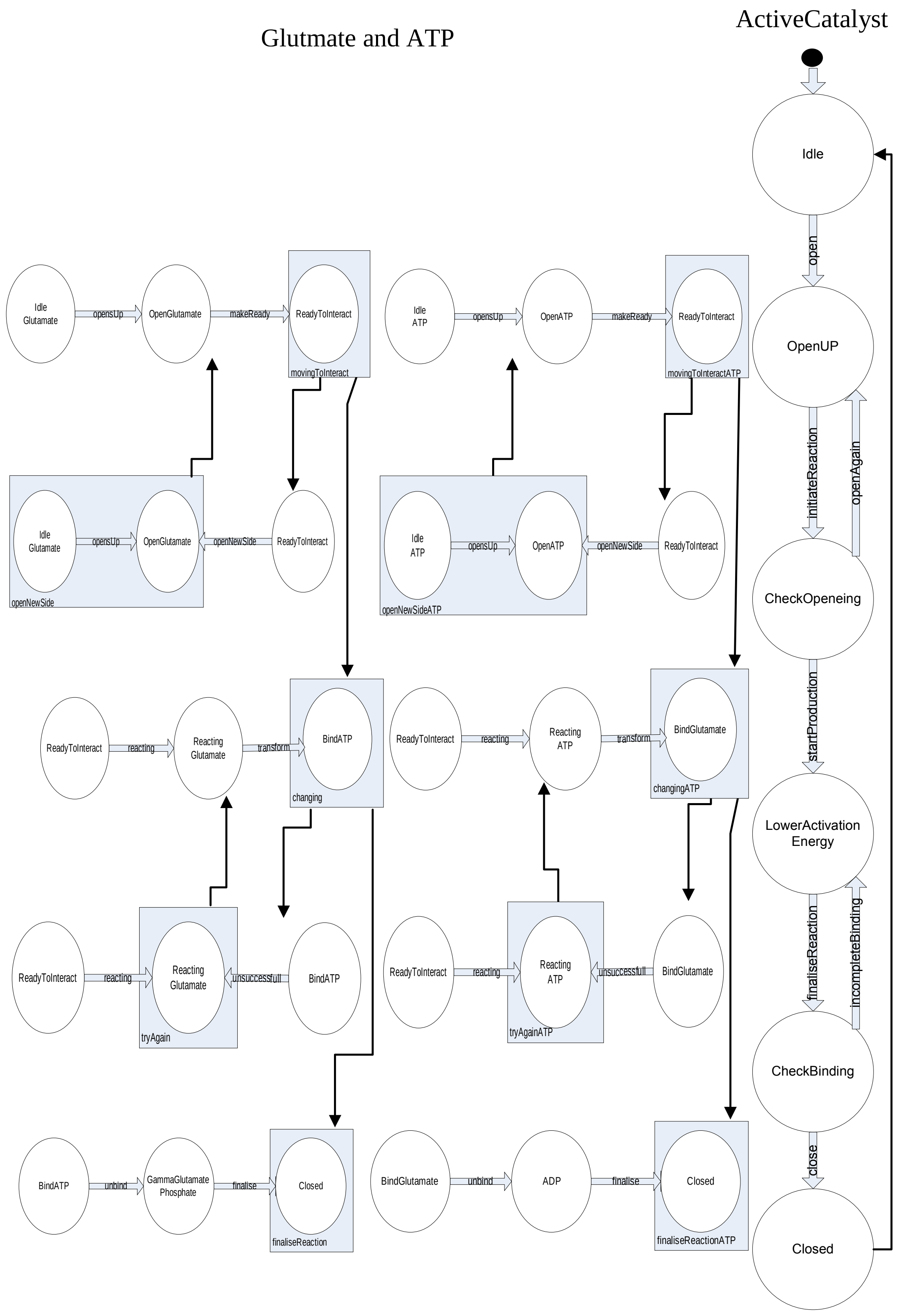


**Figure 59, Global behaviour Glutamate and ATP**

Figure 59 shows the global behaviour of the entire reaction. From this figure the different states of the ActiveCatalyst and the compounds are clear. As glutamate and ATP move on ActiveCatalyst checks their states and allows them to move from one subprocess to another via the connecting traps.

## Consistency rules

The consistency rules for this reaction would look like the following:

| | | |
|---|---|---|
| C1: Cell: Idle | $\xrightarrow{}$ | OpenUp |
| C2: Cell: OpenUp | $\xrightarrow{}$ | CheckOpening |
| Glutamate: ActiveGlutamate | $\xrightarrow{\text{movingToInteract}}$ | ActiveGlutamate |
| ATP: ActiveATP | $\xrightarrow{\text{movingToInteractATP}}$ | ActiveATP |
| C3: Observer: CheckOpening | $\xrightarrow{}$ | LowerActivationEnergy |
| Glutamate: ActiveGlutamate | $\xrightarrow{\text{movingToInteract}}$ | Reacting |
| ATP: ActiveATP | $\xrightarrow{\text{movingToInteractATP}}$ | ReactingATP |
| C4: Cell: CheckOpening | $\xrightarrow{}$ | OpenUp |
| Glutamate: ActiveGlutamate | $\xrightarrow{\text{movingToInteract}}$ | OpenWrongSide |
| ATP: ActiveATP | $\xrightarrow{\text{movingToInteractATP}}$ | OpenWrongSideATP |
| C5: Cell: OpenUp | $\xrightarrow{}$ | CheckOpening |
| Glutamate: OpenWrongSide | $\xrightarrow{\text{openNewSide}}$ | ActiveGlutamate |
| ATP: OpenWrongSideATP | $\xrightarrow{\text{openNewSideATP}}$ | ActiveATP |
| C6: Cell: CheckOpening | $\xrightarrow{}$ | OpenUp |
| Glutamate: ActiveGlutamate | $\xrightarrow{\text{movingToInteract}}$ | OpenWrongSide |
| ATP: ActiveATP | $\xrightarrow{\text{movingToInteractATP}}$ | ActiveATP |
| C7: Cell: OpenUp | $\xrightarrow{}$ | CheckOpening |
| Glutamate: OpenWrongSide | $\xrightarrow{\text{openNewSide}}$ | ActiveGlutamate |
| ATP: ActiveATP | $\xrightarrow{\text{movingToInteractATP}}$ | ActiveATP |
| C8: Cell: CheckOpening | $\xrightarrow{}$ | OpenUp |
| Glutamate : ActiveGlutamate | $\xrightarrow{\text{movingToInteract}}$ | ActiveGlutamate |
| ATP: ActiveATP | $\xrightarrow{\text{movingToInteractATP}}$ | OpenWrongSideATP |
| C9: Cell: OpenUp | $\xrightarrow{}$ | CheckOpening |
| Glutamate: ActiveGlutamate | $\xrightarrow{\text{movingToInteract}}$ | ActiveGlutamate |
| ATP: OpenWrongSideATP | $\xrightarrow{\text{openNewSideATP}}$ | ActiveATP |
| C10:Cell: LowerActivationEnergy | $\xrightarrow{}$ | CheckBinding |
| Glutamate: Reacting | $\xrightarrow{\text{changing}}$ | Reacting |
| ATP: ReactingATP | $\xrightarrow{\text{changingATP}}$ | ReactingATP |
| C11: Cell: CheckBinding | $\xrightarrow{}$ | Close |
| Glutamate: Reacting | $\xrightarrow{\text{changing}}$ | ReactionComplete |

ATP: ReactingATP $\xrightarrow{\text{changingATP}}$ ReactionCompleteATP

C12: Cell: CheckBinding $\xrightarrow{}$ LowerActivationEnergy

Glutamate : Reacting $\xrightarrow{\text{changing}}$ UnsuccessfullBinding

ATP: ReactingATP $\xrightarrow{\text{changingATP}}$ UnsuccessfullBindingATP

C13:Cell:LowerActivationEnergy $\xrightarrow{}$ CheckBinding

Glutamate:UnsuccessfullBinding $\xrightarrow{\text{tryAgain}}$ Reacting

ATP: UnsuccessfullBindingATP $\xrightarrow{\text{tryAgainATP}}$ ReactingATP

C14: Cell: Close $\xrightarrow{}$ Idle

## Consistency rules for figure 59

The consistency rules for figure 59 shows all the different states of glutamate together with ATP. These rules specify how the transitions and phase changes take place. There are no subprocess changes for the compounds as the ActiveCatalyst moves from state *Idle* to *OpenUp* therefore no rules for them have been created. Rule C6 is fired if the wrong side of glutamate is opened and rule C8 if the wrong side of ATP is opened. When glutamate becomes ready it continues with the binding process via rule C7. Rule C11 is fired when the binding is getting finalised. Rule C15 takes place when the catalyst returns to its idle state. Similarly all the other possible transitions and phase changes for glutamate and ATP are shown in the consistency rules. In the end if everything goes well the binding succeeds, glutamate and ATP transform into gamma-glutamyl-phosphate and ADP. These compounds will have their own behaviour different from the previous compounds within some other biochemical reaction.

# 6 Mozaiek

While working on my thesis I wanted to turn this project into a PhD research, since I see many possibilities with Paradigm in the biological field. For this purpose I participated in the Mozaiek programme, which is a NWO funded research grant programme. In that programme those whom are interested in doing a PhD research can apply for a grant ,which if selected gives you the opportunity to do a fully funded PhD research. In this chapter I will write about the proposal that I submitted. This will also give an idea as to why I think the use of Paradigm in biological systems modelling has great promising results.

## 6.1 Research proposal

### 6.1.1 Introduction

In biological systems the interesting and important aspects are the interactions between their components. Such components exist at different levels of description and understanding such as cells, genes, genetic networks, cells, tissues and others. To understand Biology at the system level, both static and dynamics of the entire composite structure have to be investigated. This has to be done in terms of the various components across the relevant levels, rather than in terms of static or dynamic characteristics of individual components or at just one description level. Properties of systems, such as their behavior are an important aspect, and understanding these properties could have an impact on the future study of biomedical research [8].

The important part in each system is to identify the components and the levels to be modelled and to integrate their separate descriptions on the basis of their interactions in a consistent manner. This then should result in a structural as well as behavioural description of the biological system as a whole. On the basis of such consistent integration across the various levels, of behaviour and interaction in particular, one can understand its overall behaviour in different circumstances. This requires suitable modelling of the relevant behaviours and, most importantly, interactions. It's difficult to deduce the behaviour of the entities by performing experiments. Modelling the behaviour using a modelling language in a structural and descriptional way is a much simpler and easier solution with promising results. It could still give an overall picture of the behaviour of the system without having to perform experiments.

Some approaches of modelling biological systems are based on just mathematical computer models that use differential equations to model the behaviour of the system. Dynamic modelling and analysis techniques do exist but they are based on mathematical models of biochemical networks [9]. In various fields research is being carried out in the dynamic modelling techniques. Systems have been developed that focus on exhibiting the behaviour of the entities, e.g. CAFISS. CAFISS uses Complex Adaptive Systems (CAS) to provide a way of modelling natural systems and exhibit their behaviour [10]. Similar research in fields such as neural intelligence is being carried out [11]. Statistical models are also developed and used for modelling [12]. All these systems start directly from mathematical models rather than modelling at a more global level. **These methods still lack the interactions dynamics. They do not offer a way of modelling the behaviour and the coordination between the systems consistently.** Agent based methods are also being developed for modelling. Agent based methods are based on the concept of representing biological systems by a virtual replicate. A replicate is defined as a software system which incorporates a 1:1 mapping of biological entities into software agents, with individual agents having properties similar to those of the biological entities that they represent. But this system still uses a formal mathematical model [13].

Rather than directly starting from mathematics I propose to start at a more global level. This, I believe will give the opportunity to explore different non mathematical methods in the initial stage and focus on the behavioural dynamics of the system, since directly starting from it will not give the opportunity to focus on dynamics and coordination. With differential equation-based methods it is experienced difficult to model subcellular processes. Furthermore, in differential equation-based models reuse is complicated when new details are added to the model [13]. I will start with direct diagrammatic representation of a biological structure such as a simple biochemical reaction and a genetic network. Subsequently, by following such processes and gradually adding more and more detail I will arrive at a system with structure and behavior that can be executed and observed on a computer. Later on, the system can be described explicitly by descending to a level where the application of a mathematical model is appropriate. For the understanding of behaviour my approach describes the behaviour at a global level, it is very efficient as other approaches do not provide mechanisms for the description of individual characteristics of the entities and their behaviour. For example (see Figure 60) it is not possible to model the individual interactions and binding of the protein in the reaction (indicated with 2.7.2.11).Therefore these approaches still do not offer a new way of modelling the interactions of molecules, genes or molecules etc with their environment [13].

Increasingly, there is a strong understanding by biologists that the behaviour of an individual component in a system is determined by its internal characteristics such as its state, its location and its relationships with other components in its environment [14]. There is communication and collaboration between all these entities. To predict the behavior of such systems and to model it there is a need for a modelling technique which can model all these characteristics of the entities and show the communication and collaboration taking place**. Such a technique does not exist so far which can model all these characteristics. This has provided the foundation for the need of a new approach to understanding complex biological systems**. Biologists expect that building a good dynamic model of biochemical/genetic networks is a key step towards the development of predictive models for molecules or whole organisms. Such models are regarded as the keystones of Systems Biology. They are expected to provide scientific explanations of the behaviour of biological systems in health and disease and therefore providing great assistance in fields such as molecular medicine and personalised medicine [15]. **Therefore there is a huge demand for a dynamic modelling technique which has not been developed to date**.

The Paradigm language can offer a way of modelling interactions in living systems and provide such a dynamic modelling technique. Paradigm is a behavioral coordination language introducing phase dynamic on top of detailed behavior. Paradigm is been developed at the Leiden Institute of Advanced Computer Science. I will investigate how to use Paradigm to model these interactions in terms of consistent phase dynamics and detailed behaviors and develop a system for visualization on the basis of graphical elements [16]. For the zebrafish model, I have direct access to spatio-temporal genomic data that will be used in the case studies [17] [18] [19] for which I will also use existing collaborations of Liacs with biologists (zebrafish molecular genetics).Paradigm models can be animated; the interactions can be visualized assisting a broad range of bioinformatics researchers to better disseminate their ideas so as to gain mutual understanding in commutations.

**In this proposal I wish to investigate a method of modelling biological systems in which I can model the behaviour, communication and collaboration taking place between the entities. I wish to investigate the use of the Paradigm language to model the interactions and the graphical representation and animation thereof. In my opinion this is going to result in new insight leading to more and deeper understanding of the dynamics of a system. In the end once a system for visualization has been realised I will collaborate with other biologists to test the feasibility of the system and to research to what extent a Paradigm model has lead to a new and deeper understanding of these interactions.**

### 6.1.2 Research question(s)

This research will focus on these main questions.

1. To what extent can Paradigm be used to model these interactions and how far is it compatible to modelling biological processes? If it's not compatible can other coordination languages be used such as the Unified Modelling Language (UML)?
2. Can a system be developed to visualize these interactions?
3. To what extent can a Paradigm model of interaction and visualization lead to a new and deeper understanding of these interactions?

### 6.1.3 Method/Approach

1. ***Paradigm's compatibility with biological systems:***

First it will be investigated to what extent Paradigm is compatible with modelling interactions. Starting from well described case studies as ground truth, gradually more complex situations will be addressed. As a starting point for my research small biological processes such as in Figure 63, will be used. In this figure the proB gene is being expressed into a protein which acts as a catalyst in the reaction 2.7.2.11.I shall try to model this process using the Paradigm notations. In this case the main interesting part is the behaviour and the coordination taking place between the catalyst and the protein, since the protein produced during expression controls the other reaction 2.7.2.11. It controls the products produced in the reaction and the amount. The exciting part therefore

is to see how it acts within its environment and how the whole behaviour of the system is derived from the behaviour of these two entities. This is just a preliminary example to give an idea about the dynamics of biological processes and to explain the use and role of Paradigm in modelling such processes. I did not show the zebra fish model since that model can be quite complex.
Our case studies will be using data drawn from the zebrafish system. In the past years a considerable pool of data/networks for zebrafish has been developed within our collaborations with biologists [18] [19]. This basic simple process is a good starting point for modelling. Once I am able to model it using Paradigm, the next focus will be on more complex examples. Starting from such simple examples the same process can be applied to much more complex data such as the zebra fish model. The zebrafish model is good for comparison and validation of my Paradigm based model of the same zebrafish model .A lot of research has been done on the zebra fish model and results are known. **Using that as a starting point I could use it for validating my results. This could give me an estimate of the validity of Paradigm and how well it can be used for modelling.**
For the modelling of the systems I will make use of Rational Rose [20] and BioUML [24], both are standard software used by computer scientists and biologists. They provide a static way of visualizing biological systems. I will use Paradigms concepts in these software systems and try visualizing biological systems. How to make it visually more dynamic will be the focus of my research in the next step.
In the unlikely case the Paradigm language turns out not to be sufficient for the approaches proposed the concepts of phase dynamics will be adapted to modify other modelling languages such UML. UML and Object oriented approaches have been successfully applied as a cell and biochemical modelling languages [21]. However, they do lack the behavioural dynamics, but a system could be built on top of it which focuses on the behavioural dynamics and coordination.

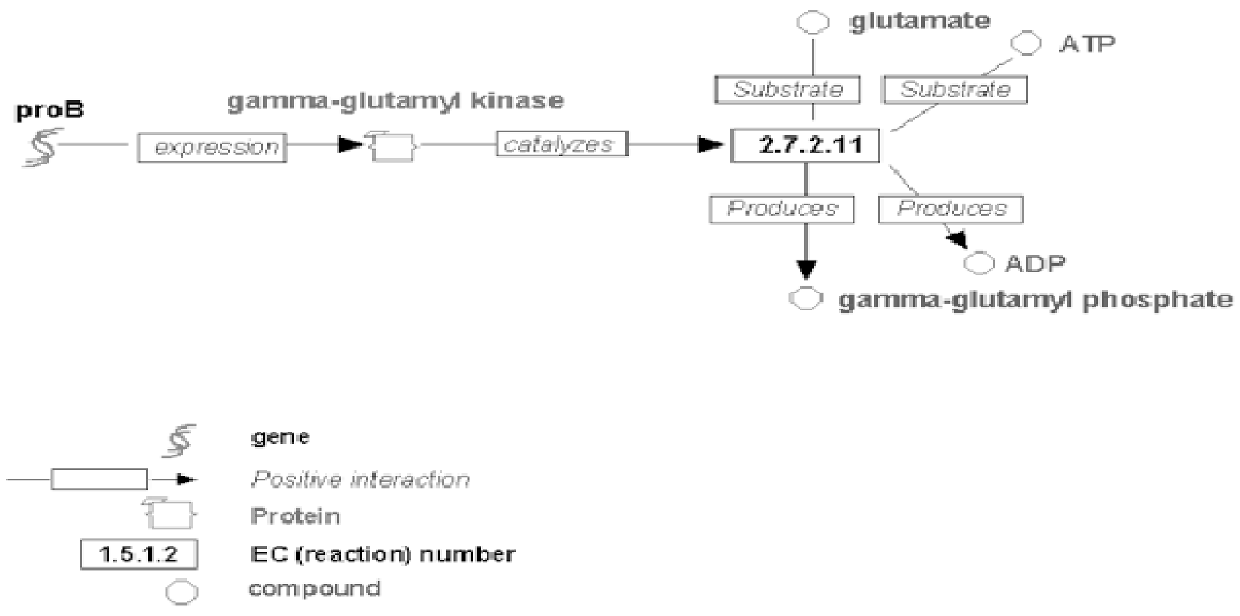


**Figure 60, Expression of proB gene into a protein [7]**

2. ***Software for visualization:***

Once I am able to model complex biological systems my next focus will be on developing a system for visualizing this. A lot of programs have been developed which I can use as a starting point in my project**.** These software systems are used by biologists and their good standard software. Which system is the best for my project has yet to be researched. The following are a few:

-Chilibot: It searches PubMed literature database about specific relationships between proteins, genes, or keywords. The results are returned as a graph. It supports several different search methods [22].

-BioPAX: It is a common exchange format for biological pathways data [23].

-GenMAPP : An academically based organization that develops and supports GenMAPP (Gene Map Annotator and Pathway Profiler), a computer application designed to visualize gene expression data on maps representing biological pathways and groupings of genes [4].

-BioUML: Is a java framework for Systems Biology. It includes access to databases with experimental data, tools for formalized description of biological systems structure and functioning, as well as tools for their visualization and simulations [24].

-Chalkboard: A prototype tool for representing and displaying cell-signaling pathway knowledge, for carrying out simple qualitative reasoning over these pathways and for generating quantitative biosimulation code [25].

All these systems can be used in developing a system that visualizes the dynamic behavior. These systems provide a static way of visualizing biological systems. A more dynamic model could be built on top of it.

3. ***Contribution of Paradigm to a deeper and better understanding of biological systems:***

For the onset of the project collaboration with biologists will be established. This collaboration will be intensified once a system for visualization has been developed (first year). However, usability and evaluation testing of prototypes of the visualization system will also be done in close collaboration with potential users. This will help to increase the usefulness of the system for biologists and detect potential flaws. For the purpose of case studies I will strongly benefit from the existing collaboration of Dr. Ir. F. J. Verbeek with Prof. Dr. H.P Spaink (Molecular Cell Biology) and Prof. Dr. M.K. Richardson (Integrative Zoology) with the, Institute of Biology (IBL) of the Leiden University. Existing local resources of micro array data as well in-situ hybridizations of the zebrafish model will be explored to detail within this collaboration. Moreover, research networks such as ZF-models and Smartmix will be addressed [26]. In this matter we will obtain an idea as to how far Paradigm has lead to a better and deeper understanding of biological systems

### 6.1.4 Innovation

A technique for dynamic modelling which can visualize interactions between the entities in a biological system does not exist to date. Bioinformatics is a multidisciplinary field involving people of various fields such a mathematics, computer science and Biology. Coordination and cooperation between these researchers is important and is needed in various projects. I expect a system modelled with Paradigm to be easier to understand and to comprehend for people in a multidisciplinary setting on the basis of visualizations,

rendering the relevant dynamics. Paradigm offers a completely different way to visualize coordination and behaviour of the components. It has its own mathematical specification of global phase dynamics, defined consistent with detailed behaviour. It uses certain selected and somewhat adapted UML-like (Unified Modelling Language) constructs to visualize component behaviours, detailed as well as global. The key difference between UML and Paradigm is, where UML's specifications of behaviour and interaction are substantially lacking in behavioural consistency, Paradigm does offer such consistency. So the visualizations are based on solid specifications and will be more convincing. This is expected to result in more thorough understanding and better cooperation between people of different backgrounds.
Finally such a dynamic modelling technique is expected to provide essential support in various fields such as molecular Biology or medicine, since ideas can be conveyed in a structured way leading to a better understanding.

### 6.1.5 Relevance for science, technology or society

I believe my research will provide a better way of understanding complex interactions that take place in living systems before any other modelling technique is applied; further it will assist in predicting the behaviour of individual entities in living systems such as cells, genes, and pathways. Finally, graphical and animation tools will make a significant contribution in visualizing what happens and how it happens; in this manner the problem can be conveyed more clearly. Well described models are regarded as the keystones in Systems Biology. Using the approach that I propose it will result in models with well described behavioural dynamics and therefore they will support scientific explanation and also provide a better link between model and the practice in the biomedical field. My approach will allow conveying ideas in a structured and comprehensive manner.

# 7 Learning Process

During the course of writing this thesis I learnt many new things. In the first stage I had to read about Biology to get a basic understanding of how the processes work. Making the first step on how to combine biological processes with Paradigm was complex. I spent too much time on trying to explore every detail of the biological process, when a certain molecule opens or how and when it attaches itself to some other molecule etc. I thought that this could give a clue about how I could use Paradigm in modelling its behaviour. This was not a good idea since the overall behaviour of the molecules is known, such detail as when it attaches or when exactly it opens up was not important. Biologists probably already know this. I needed to apply Paradigm in modelling the behaviour in such a way that it could give a different view on how the process is taking place and how its overall behaviour is affected/changed by the behaviour of other molecules around its environment. This in my view would be where Paradigm could play its role and show a different way of modelling processes than what is traditionally used by systems biologists and Bioinformaticians. Once this was clear, the combination of Paradigm to modelling biological processes was easily understood.

During the second stage of writing this thesis which was when I started with modelling, the most important thing I learnt was the new insights gained during model building. During the modelling of these processes it happened many times that after a model was made, a new look or insight into the process was achieved. While modelling the overall global behaviour of a process, new things about its behaviour like how it could change in more different ways or how it could be affected due to the interference of some other entity used to emerge. This would give me a better understanding about the detailed behaviour, and new processes that were important for the detailed behaviour. In this way a more better and detailed model, modelling the behaviour of the entire process was achieved taking every detail into account.

# 8 Discussion and Conclusion

For building any model or simulation of a molecule two separate architectures can be taken into account. The first is the top-down containment structure – modelling from the top such as the membranes going to the bottom such as the small molecules. The other architecture is the bottom-up behaviour – starting from the bottom modelling the dynamic reactions between molecules, and the rules and parameters that define these reactions going to the top to the membranes. UML makes a fundamental distinction between structural modelling and behavioural modelling of computer systems and has been started to be used to a limited extent by the biological community for building biological systems. The biological community is starting to use concepts from computer science and biology together for biological systems modelling and is looking at alternative methods for modelling and simulating molecular and biological processes than the ones traditionally used. Traditional methods used lack the techniques for modelling the dynamic behaviour of systems. Therefore there still is a huge need for a dynamic system which facilitates the modelling of the behaviour, communication and coordination between the entities in a system. This thesis used the bottom up approach since it offered mechanisms to deal with these types of architectures using Paradigm together with various concepts from UML.

## 8.1 Discussion

### 8.1.1 Paradigm and traditional methods

In Paradigm, reaction modelling is implemented using message passing, the manager and employees communicate by passing messages while moving from one state to the other. Interactions between biological elements are therefore, directed as messages are passed from one active object to another. This makes it possible to model very detailed, localized phenomena in the systems such as a cell, which are extremely difficult to capture in a differential equation representation. Paradigm separates behaviour and interaction while traditional methods do not.

Paradigm together with UML can provide a simple class-like reuse mechanism where the concepts of classes and inheritance can be used. None of the biological modelling tools provide a concept of subclass, or of a multiplicity of objects of the same type. This can be of huge importance when building a simulation of the system. The Object Orientation concepts together with UML make use of classes, static and dynamic instantiation of objects from classes, subclasses, and multiplicity. This concept of classes can be of importance when building a simulation since entities can be dynamically reconfigured (connected to each other, disconnected, and reconnected to other entities). This makes running and building a simulation a lot easier.

### 8.1.2 Paradigm's usability with biological systems

The first step during this research project was to investigate if Paradigm is usable for modelling biological systems. For this purpose I looked at other behavioural coordination languages. Having researched about it, I found out that UML has been successfully applied in modelling biological systems. Paradigm is very much similar to UML. The key difference between UML and Paradigm is, where UML's specifications of behaviour and interaction are substantially lacking in behavioural consistency, Paradigm does offer such consistency. Therefore since UML has been successfully applied, I expected no problem with Paradigm either and in the worst case could use UML as a backup if Paradigm was not successful. This was not needed in the end since Paradigm proved to be compatible for modelling, but I did use some concepts of UML such as aggregation diagram and coordination diagram along with Paradigm for greater potential in modelling flexibility.

Modelling systems where only two or three entities are involved with Paradigm was comparatively simpler than modelling a larger process which involves a multiple of entities influencing each others behaviour. The complex part was developing the understanding of the biological process and thinking of all the possible transitions, communications, coordination and processes. With no background in biology I did not know how the entire process is taking place and how an entity is being influenced by the other. Once some basic understanding of biology was gained this task became a lot easier.

The binding of the molecule was assumed to be a simple example but it took a lot of time getting familiar with Paradigm and the biological details. Translating the rules and concepts of Paradigm into a biological model of a system is a difficult step. Paradigm is still in development and is very conceptual so it’s difficult translating those rules into a working model and getting familiar with Paradigm. Paradigm is still in development and its concepts keep changing. During the start of this thesis the concept of a manager was considered to be a requirement in the models since coordination is achieved in terms of coordination between manager and employee. This concept has changed and a manager is not needed for achieving coordination between components involved, the components themselves can achieve this now. I used the concept of a manager in the models but now since a manager is not needed the models could have been modelled totally differently. This is restricting in terms of having all these rules to adhere to during the initial phase of modelling. With a basic background in biology and good understanding about Paradigm modelling can become relatively easier.

Modelling a larger process could take a lot of effort and time, since the process can be modelled in great detail. With larger processes it would be difficult if there are many processes that are interconnected, influencing each other’s global behaviour. If the number of scenarios that have to be taken into account is many it will increase the number of subprocesses automatically. This can make the model very large with the number of consistency rules and the global behaviour diagrams. With no visualisation it can take a lot of time modelling such large systems. The models on the other hand do give a very good idea about how the entities are influencing each others behaviour and if something goes wrong with one entity how that will effect the entire system. Working closely with biologists and having the models checked for their consistency each time would make modelling complex processes a lot easier.

The kind of behaviour biological reactions exhibit, the most important thing about them is the interactions between the entities involved and their environment. Having read more about Paradigm and its concepts, it became clear that Paradigm can be very compatible for this purpose since Paradigm stresses on the dynamic behaviour of the entities and modelling the interactions and changing’s taking place. I used two preliminary examples to give an idea about the dynamics of biological processes and to explain the use and role of Paradigm in modelling such processes. Paradigm has been applied successfully in modelling these two examples. These were relatively small examples with only two or three entities involved. Modelling a more complex example using Paradigm is an open issue.

### 8.1.3 Contribution of Paradigm to a deeper and better understanding of biological systems

The visual diagrams in Paradigm make the generated models far more accessible to such to people with various backgrounds such as Computer Science and Bioinformatics. Physical structures are naturally represented in a global behaviour diagram and interactions are explicitly represented using the concepts of states and transitions between the states. The alternative differential equation representation hides interactions in terms of equations, which are significantly more obscure for the non-mathematician. With interactions "hidden", it makes model reuse more difficult.

Bioinformatics is a multidisciplinary field involving people of various fields such as Mathematics, Computer Science and Biology. Coordination and cooperation between these researchers is important and is needed in various projects. Models built with Paradigm are easier to understand and comprehend for people in a multidisciplinary setting on the basis of visualizations, rendering the relevant dynamics. With a Computer Science background, for me the models that biologists use are difficult to comprehend. Paradigm's visualizations are based on solid specifications therefore they are more convincing. I expect this to result in more thorough understanding and better cooperation between people of different backgrounds. It needs to be researched if Bioinformaticians and systems biologists agree with this. Once some standard software is available for modelling, end users could be asked to evaluate it and this aspect could be researched. This way it could be investigated if users find Paradigm easy to use or not and whether it really results in better understanding of biological systems.

## 8.2 Future work

I used two examples for modelling; in the future more complex data could be modelled. For this purpose it is very important to collaborate with system biologists and Bioinformaticians. This would be important for usability and evaluation testing of prototypes of the visualization system, since Bioinformaticians or systems biologists are the ones whom will be using this modelling technique in the future. In this way a better idea could be obtained as to how far Paradigm has lead to a better and deeper understanding of complex biological data. Therefore it could be used very successfully to validate the results and conclude if Paradigm is compatible for modelling complex data or not.

### 8.2.1 Software for visualization

Biologists use modelling software such as BioUML and GenMAPP. I used GenMAPP, using Paradigm's concepts in it to model biological reactions. This was done to get familiar with biological software and see if it could be extended into dynamic software. This software system provided a static way of visualizing biological systems. I used

Paradigms concepts in it and tried visualizing biological reactions more dynamically. This was done quite successfully since the models I have built in chapter 4 were built using this program. There is no standard software for Paradigm yet to visualize its concepts diagrams and this program could be used as a starting point for building dynamic software. It would be a lot easier if some standard software was available, since it would save all the time using different software together. How to make it visually more dynamic can be the focus of future research.

A lot of programs have been developed which can be used as a starting point for building a dynamic visualisation system. I researched about the following;
-GenMAPP: An academically based organization that develops and supports GenMAPP (Gene Map Annotator and Pathway Profiler), a computer application designed to visualize gene expression data on maps representing biological pathways and groupings of genes.
-BioUML: Is a java framework for Systems Biology. It includes access to databases with experimental data, tools for formalized description of biological systems structure and functioning, as well as tools for their visualization and simulations.
-PetriNets: A formal graphical mathematical modelling language used for the representation of discrete-event dynamic systems. It depicts the structure of distributed systems as directed graphs.

GenMAPP is good for biological pathways building and BioUML offers a lot of functionality as well. The problem with BioUML is that it is not stable therefore I did not use it for building the STD's, while GenMAPP is more stable. Many software systems are available to visualise PetriNets such as Arp and CoopnBuilder. These software systems can be good starting points for building dynamic software on top of them visualising Paradigm's concepts in it. GenMAPP is an open source project and provides a good foundation for building dynamic software which can visualise Paradigm's concept. Right now it can only be used for building STD's but it could be extended to include the different UML and Paradigm diagram types which were used in this thesis and offer functionality for making consistency rules. This would make model building a lot easier and would save a lot of effort and time spent on building the models.

### 8.2.2 Validation and simulation

Once a dynamic modeling tool based on Paradigm specifications is built the next step is a simulation tools based on Paradigm notations which is needed for non-mathematicians using visual modelling and programming in order to increase the accessibility of modelling in biology. Paradigm together with UML and OO (Object Orientation) concepts can be extended to a system which facilitates building simulations of the systems. Rational Rose Software has been used for building the UML diagrams. In this system programming code can be added behind the diagrams. That way the entire system can be simulated. Paradigm diagram types such as a global behaviour diagram cannot be simulated like that in Rational Rose, since its not part of the system. Therefore it remains to be researched whether Rational Rose could be extended into a dynamic system which includes Paradigm's concepts. It's important to have a system which facilitates building simulations of the models. These simulations can be used to validate the models. Rational Rose is not freeware therefore it is difficult to extend that into Paradigm diagram types. Another tool called Gepasi can be used which does claim to produce accurate quantitative results. In addition to the practical value of having Paradigm generate accurate results, Gepasi could help to validate its design and implementation. Gepasi with its roots in biochemistry and differential equation modelling, focus primarily on the moment-by-moment time evolution of the behavioral architecture, but also provides varying amounts of support for aspects of the complex structural architecture. [27] [28] [29]

## 8.3 Conclusion

I believe Paradigm can be successfully applied in the biological research field since it provides a good way of understanding complex interactions that take place in living systems before any other modelling technique is applied. This can assist in predicting the behaviour of individual entities in living systems such as molecules, genes, and pathways. It remains to be researched if Paradigm is equally usable with complex data, but for modelling smaller biological systems Paradigm has proved to be usable. As a modelling language Paradigm provides a good foundation for modelling interactions in living systems. Interactions are explicitly represented in Paradigm using the concepts of states and transitions between the states. This makes model reuse easier. The Systems Biology community and Bioinformaticians can benefit a lot from models built with Paradigm since it provides a good technique for modelling the dynamic behaviour of systems. Models in Paradigm can be modelled at a more global level and offers a good foundation for modelling the behaviour and the coordination between the systems consistently. The traditional methods lack such techniques. The visual diagrams can make the models accessible to people with various backgrounds since physical structures are naturally represented.

Paradigm is still in development and its concepts keep changing, this changed how models could have been modelled and the behaviour shown of the entities by such models. I used the concept of a manager in the models but now since the concept of a manager is not a requirement the models could have been modelled differently. This is restricting in terms of having all these rules to adhere to during the initial phase of modelling. Paradigm as a modelling language is very conceptual and has many rules to adhere to; translating those rules into a model is a difficult step. The rules and syntax are difficult to understand for a beginner and applying those rules into modelling biological systems is a difficult step. Making the first step therefore is very difficult. Modelling smaller data was harder than anticipated due to these setbacks and took a lot of time. The model can get very large as the number of scenarios increases, therefore modelling a larger process could take a lot of effort and time.

There is no visualisation tool to visualise Paradigm’s concepts. Dynamic graphical and animation tool will make a significant contribution in visualizing what happens and how it happens; in this manner the problem can be conveyed more clearly and will simply the process of modelling.  Since there was no visualisation tool it was difficult and time consuming to use different software together for building the models. A lot of time was spent on investigating which program was suited for this purpose. A system which facilitates building biological models using Paradigm is needed together with a simulation tool.

# 9 Acknowledgements

I would like to thank my supervisors, Dr. Ir. Fons Verbeek and especially Dr. Luuk Groenewegen whose patience and support during the process of modelling was invaluable. My family and friends whom supported me; their support and encouragement have made all of this possible. Last but certainly not the least; I would like to thank Monique for her help in explaining the biological processes and checking the models for their consistency and Laura for her help in the layout of this thesis.

**Thesis supervisors:**

**Dr. Ir. Fons Verbeek**
Bio-Imaging Group, LIACS, Leiden University

**Dr. Luuk Groenewegen**
Software Engineering & Information Systems Research Group, LIACS, Leiden University

# 10 Appendix

## 10.1 Research proposal

In this thesis I will introduce a new method of modelling biological systems in which you can model the behaviour, communication and collaboration taking place between the individual entities in a system. I will investigate the use of the Coordination language Paradigm to model the interactions and the graphical representation and animation thereof. This I believe is going to result in new insight leading to more and deeper understanding of the dynamics of a system.

The Paradigm language can offer a new way of modelling interactions in living systems. Paradigm is a behavioural coordination language developed at the Leiden institute of advanced Computer Science. Like most behavioural coordination Languages Paradigm has so far been applied in modelling business models or software components. The idea of using Paradigm in modelling biological systems therefore is new and has not been investigated so far. I will investigate how to use Paradigm to model these interactions in terms of consistent phase dynamics and detailed behaviours and see if a system for visualization can be developed on the basis of graphical elements.

Using a case study of a simple biological system as a starting point, Paradigm will be used to see how this system can be modelled and whether this is compatible or not. Gradually I will move to a more complex example and see if it’s equally compatible. The behaviour of an individual component of a system is determined by its internal characteristics (state), its location (place) and its relationships with those components around it (communication). This is a dynamic process, so the modelling can be done according to each case like a cell or a molecule interacting with each other and with their environment.

## 11 References

[1] Hucka, M., et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19, 524-531, 2003.
[2] Reference about UML and Keuster: http://wwwcs.uni-paderborn.de/cs/ag-engels/skripte/publication/pub_list_eng.php?AuthorID=4
[3] L. Groenewegen, N. van Kampenhout, and E. de Vink ,Delegation Modelling with Paradigm, (Eds. J. M. Jacquet and G.P. Picco,), *Proc. Coordination 2005, pages 94-108, LNCS 3454, 2005.*
[4]GenMAPP , http://www.GenMAPP .org/introduction.asp
[5]L. Groenewegen, J. Stafleu, and E. De Vink, Coordination and On-the-Fly Migration as Constraint Orchestration, june 5 2007.
[6]http://www.brc.dcs.gla.ac.uk/~drg/courses/bioinformatics_mscIT/slides/slides9/sld009.htm
[7]http://www.cs.tau.ac.il/~rshamir//algmb/00/scribe00/html/lec14/node3.html#lec14:Fig:pi1
[8]Hiroaki Kitano,Systems Biology: A Brief Overview, http://www.sciencemag.org
[9] Georgy P.K., Artem S.N. and Eugene V.K., Mathematical modelling of tumor therapy with oncolytic viruses: effects of parametric heterogeneity on cell dynamics, *Biology Direct, Oct 3 2006*
[10] Joc C.T and Atul J., CAFISS: A Complex Adaptive Framework for Immune System Simulation, *ACM Symposium on Applied Computing. 2005*
[11] Zak M,Creative Dynamics approach to neural intelligence, *Biological Cybernetics, 1990.*
[12] Paola G., Elisa G. and Ester P., Statistical external validation and consensus modelling: A QSPR case study for Koc prediction,*Journal of Molecular Graphics and Modelling, 4 August 2006*
[13] Holcombe M, Smallwood R, Agent-based modelling in Biology: From molecules to population, *II International conference on computational bioengineering, H. Rodrigues et al. (Eds.) Lisbon, Portugal, September 14-16, 2005*
[14] Khan, S., A Multi-Agent System for the Quantitative Simulation of Biological Networks. *AAMAS'03*, 385-392 ,et al., 2003.
[15] www.dcs.shef.ac.uk/~rod/Integrative_Systems_Biology.html
[16] Natal A.W van Riel, Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments, *Briefings in Bioinformatics, vol 7 No 4, 2006*
[17] Belmamoune, M. and Verbeek, F.J., Heterogeneous Information Systems: bridging the gap of time and space. Management and retrieval of spatio-temporal Gene Expression data, InSCit2006 (Ed. Vicente P. Guerrero-Bote), *Volume I "Current Research in Information Sciences and Technologies. Multidisciplinary approaches to global information systems", pp 53-58. (2006)*
[18] Meuleman, W., Welten, M.C., Verbeek, F.J. *Construction of correlation networks with explicit time-slices using time-lagged, variable interval standard and partial correlation coefficients*.Lecture Notes in Computer Science, Volume 4216, Computational Life Sciences II, pp 236-246. (2006)

[19] Corredor-Adámez M., Welten, M.C.M. , Spaink, H.P., Jeffery, J.E. ,Schoon, R.T. ,de Bakker, M.A.G. Bagowski, C.P., Meijer, A.H. , Verbeek, F.J. and Richardson. M.K. (2005).*Genomic annotation and transcriptome analysis of the Danio rerio hox complex with description of a novel member, hoxb13a*.Evolution & Development 2005 (#5):362-375
[20] http://www-306.ibm.com/software/rational/
[21] Webb K,White T, UML as a cell and biochemistry modelling language, *Biosystems, Vol. 80, No. 3., pp. 283-302, June 2005*
[22] www.chillibot.net
[23] BioPAX: Biological Pathways Exchange Language Level 2, www.biopax.org
[24] BioUML, http://www.biouml.org
[25] Cook D.L., Wiley J.C. and Gennary J.H.,Chalkboard: Ontology based pathway modelling and qualitative inference of disease mechanisms, *Proceedings of the Pacific Symposium on Biocomputing*, pp. 16-27, *2007*.
[26] http://bio-imaging.liacs.nl/ZebrafishInformatics.html
[27] Mendes, 2003. Gepasi 3.30. Available via the World Wide Web at http://www.gepasi.org.
[28] Mendes, P., 1993. GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems. Comput. Appl. Biosci. 9, 563-571.
[29] Mendes, P., 1997. Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3. Trends. Biochem. Sci. 22, 361-363.