Exploring signalling pathways as a serious game

Brend Wanders

December 14, 2011

Master's thesis *"Exploring signalling pathways as a serious game"*
University of Twente – December 14, 2011(revision 194)

Author:
Brend Wanders*(s0088897; b.wanders@student.utwente.nl)*

Supervisors:
Dr. Paul van der Vet,
Dr. Ir. Rom Langerak,
Ir. Jetse Scholma.

Abstract

INAT is a tool for the interactive analysis of signalling pathways dynamics in living cells developed at the University of Twente. This report presents an analysis of the tool's usage scenario's to determine which features are good directions for further investigation. Based on the analysis a reimplementation of the tool's front-end in the biological workbench Cytoscape is presented. Finally, continuing on the analysis, the design for model element annotations is presented. The annotations are intended to take the role of lab journal for in-silico experiments.

Contents

Chapter 1

Introduction

The research presented in this report was done as a master's thesis within the Human Media Interaction master programme at the University of Twente. The research continues upon Wim Bos' work on INAT [4].

The continuation of work on INAT by myself and Stefano Schivo coincided with the start of a multi-disciplinary group of colleagues and students, the Executive Biology group. At the time of writing the executive Biology group consists of: Ricardo Urquidi Camacho, Marcel Karperien, Rom Langerak, Jaco van de Pol, Janine Post, Stefano Schivo, Jetse Scholma, Paul van der Vet and Brend Wanders.

Concurrent with my work, Stefano Schivo has been working on the formal models supporting the tool and a paper is in the making by this group, primarily written by Stefano Schivo and Jetse Scholma, with contributions from the whole group. As the tool is close to being published, and therefore publicly known, the group has decided to rename the tool 'Animo'.

1.1 Previous Work

1.1.1 INAT

Wim Bos' tool INAT allows users to model signalling pathways. Such a model is an abstraction of the biological signalling networks that are active within living cells. The behaviour of signalling networks can be simulated, producing in-silico measurements and allowing the user to inspect the simulation results. The simulation is based on the formalism of timed automata.

Signalling pathways are one of the methods used by cells to communicate. A signalling pathway is based on a chain of reactions that either activate a protein or inhibit activation of a protein. A protein's activation is affected by a catalyst, the activated protein will then act as a catalyst for the next step in the chain and so on. Some of the steps in the chain may act as inhibitors, preventing or slowing down the activation of the protein they inhibit. More information on signalling pathways can be found in [1].

Timed automata are finite-state machines extended with channels and clocks. Clocks are used to enable or disable certain transitions based on conditions relating to time. Channels allow the synchronisation of multiple timed automata by enabling certain transitions only when both automata make use of the same channel. More information on timed automata can be found in [2].

The INAT tool offers basic interaction and operations. It allows for the creation and modification of networks with simple reaction and reactant parameters. The tool can display the results of a simulation by letting the user create slides of the state of the network at certain points in time. It is also possible to export a movie that displays network state over time as an animated sequence.

INAT leverages UPPAAL [13] to simulate the models. UPPAAL as a state-of-the-art tool for the analysis of timed automata. Translation from the user-drawn model in INAT to the formal model required by UPPAAL is done by INAT when a simulation is requested by the user, so now knowledge of UPPAAL or the formalism is required from the end-user.

1.1.2 Other tools

Although this work is mainly based on INAT, it is important to be aware of other efforts[1]. There are many different ways to model biological systems, each with its own strong points. It is possible to distinguish among modeling approaches by looking at which formal method they employ.

By doing so, we arrive at two large groups: one includes approaches based on ordinary differential equations (ODEs), while the other encompasses all methods based on concurrent systems. This distinction captures the main peculiarity of concurrent systems, which allows one to describe a system by specifying its components in isolation, and then define the interaction rules; on the other hand, a method based on ODEs will need to explicitly list and precisely define every possible interaction between the various components of the system.

Models based on ODEs are usually easier to understand, because of their strong connection with the actual chemical reaction laws which govern the evolution of a system, while an approach based on concurrency will usually add a layer on top of the canonical description of chemical reactions, requiring the user to acquire some practice before being able to fully profit from the paradigm.

Tools that allow models directly based on ODEs include COPASI [16], E-Cell [25] and GNA [8]. Among these tools, COPASI in particular allows the user to define models based not only on ODEs, but also on a hybrid between ODEs and stochastic models. The employment of stochastic dynamics is useful in those cases in which the number of molecules of a chemical species is particularly low, and the use of a purely analytical method can lead to unexpected results [10]. The E-Cell tool also allows the user to define different types of simulation approaches, so that the model can be implemented as a stochastic process, using ODEs, or via Differential-Algebraic Equations.

[1] This section is largely taken from the paper in the making. Some parts have been rephrased, restructured or removed altogether to fit this report.

Methods relying on concurrent systems, such as BlenX [9], Bio-PEPA [6], Cell Illustrator [18] or Statecharts [12], generally allow the user to define the features of each entity involved in a model, and then specify the ways in which these entities can interact. Such methods can be divided into two subgroups: qualitative and quantitative, the distinction being the possibility to add to the model numerical (quantitative) information such as speed of reactions, concentration of species, volume of solution.

INAT places itself among the quantitative methods based on concurrent systems. Differently from many tools in the same class, INAT allows the user to decide on specific levels of precision and granularity for each part of the modelled network.

Nevertheless, INAT has its downsides: for example, it has currently no support for widespread modelling languages and formalisms such as ODEs or SBML, both of which are supported by BlenX, Bio-PEPA and E-Cell, among others. These tools allow the user a possibility to employ more than one single formalism, exploiting the strong points of each one.

Moreover, the graphical output capabilities of INAT still have a margin for improvement: while it is possible to observe the state of the system at any point in time in a colour-coded fashion directly on the modelled network, the animation capabilities of COPASI, Cell Illustrator or Statecharts offer more possibilities.

1.2 Research Focus

This research explores how the INAT tool can continue its evolution to become a serious game. To this end, the following question has been asked: *"Now that we have this model, what do we want with it and what can we do with it?"*.

To continue development into a serious game, we need to have a playing field, we need to know what moves to make available, and we need to know what kinds of bookkeeping are necessary to have the user benefit from playing our game.

In this work, the playing field is represented by the front-end of the tool, as the front-end is the way the user views the model and the front-end is used to issue commands. The moves are represented by the actions the user wants to take, and the effect these actions have upon the model or the state of the tool itself: in short, the user interaction with the tool. The bookkeeping should offer accumulated information to the user, and allow her to look back on her work and present it to others without the need to add additional documents or do manual bookkeeping.

Chapter 2

Analysis & Design

This chapter presents six usage scenario's and analyses them to determine what features would make strong improvements. These features need a solid framework in which they can be implemented. Following the usage scenario's is the design of tool's integration with Cytoscape. Cytoscape [21] is a well-known and commonly used biology workbench focused on visualisation. The chapter is concluded by the design of annotations.

2.1 Scenario's

This section presents the analysis of the tool's intended usage. The first part of this section, 2.1.1, contains a description of those features that are important in all scenario's. The rest of this section contains the analysis of the usage scenario's. Though the analysis of usage scenario's follows directly after the generic features, the Cytoscape integration (see 2.2) enables future implementation of the features in this section.

The analysis of usage scenario's is both a result of this research, and a starting point for further work. The scenario's are discussed from the viewpoint of one or more users of the tool. Six usage scenario's have been defined:

- Creation: creating a new model from scratch.
- Refinement: adapt an existing model to fit new information.
- Collaboration in meetings: multiple users discuss and modify a model.
- Review: peer-review or progress review of a model.
- Exploration: playing around with the model to see how it behaves.
- Integration: combining multiple models into one.

Each scenario gives rise to a few desired features of the tool. Some features may have some overlap in other scenario's, but have been categorised with the scenario that benefits from them most.

2.1.1 Miscellaneous

Some features are desirable, but not clearly defined by any one of the usage scenarios. The features presented in this section are important in all scenario's.

Visualisation of information The tool contains large amounts of information, too much to visualise at once. Since not all information is relevant at all times the tool should allow the user to determine what information she wishes to be visualised, and what information to display only on demand.

The tool should allow the user to store and recall such visualisation settings. This allows the user to define multiple views of the model to be selected at will. Care should be taken to allow the user to quickly customise these views on the fly, and still retain her personal presets.

A last option would be to allow the exchange of views, allowing multiple users to use the same view when collaborating.

Works like [27] and [20] offer a starting point for further work on visualisation.

Selection strategy The selection of model elements is a vital part of the tool, as most operations are only available once the user has indicated on which elements she wants to operate. Users of the tool do not consider each element apart, they group model elements in semantic divisions.

The selection strategies offered by the tool will have to match these divisions of the model to allow the user to quickly indicate those elements of interest to her. To this end, the model should allow at least the following selection strategies:

- Upstream reactants: adds all reactants upstream of the current selection,
- Downstream reactants: adds all reactants downstream of the current selection,
- Upstream reactions: adds all reactions upstream of the current selection,
- Downstream reactions: adds all reactions downstream of the current selection.

Further testing should be done to see if the users frequently use other selection patterns, and whether they wish to filter their selection to narrow it down.

Group operations During alterations of the model a lot of parameters have to be set. This can easily become tedious manual work.

The tool could assist the user by allowing her to set common parameters for all applicable elements in the selection. By showing only those parameters common to all selected elements in the parameter area of the tool, the user can quickly see which variables she can edit for the whole selection.

Copy & Paste Users have come to expect programs to offer extensive copy and paste functionality, reducing amount of work involved in the duplication of patterns.

Care should be taken to make sure that the tool lives up to the user's expectations in this regard. Model elements should be copied with a complete set of parameters and meta-data like relative positions intact. If at all possible, the copied model elements should carry with them some means of identifying their source.

2.1.2 Creation

The creation scenario puts a user to the task of creating a model from scratch. Creating a model from scratch is hard, so the user should be able to access other data sources.

Even if the user does not create the model from scratch, the features described here allow her to take advantage of external data sources and allow the tool to assist in the extension of existing models.

Published pathway importing Published pathways are pathways on which a lot of data has been published. Some of these are widely accepted, while others are considered tentative. A lot of information is available about these published pathways.

Allowing the import of published pathways allows quick creation of a base model. After import of a pathway, the user can change the pathway by extending it, or cutting out the non-relevant parts. Some possible sources for import are: a public database (see 2.1.2), SBML files, research group repositories, and personal archives.

The tool should allow the user to extend her own personal archive with pathways she created herself. The tool should allow the user to add a short description to any pathway she adds to her personal archive.

Adding new pathways to the personal archive carries the danger of making unproven or unverified pathways appear as accepted fact. This danger is mitigated by the fact that the tool is meant for exploration, and the user can add a description to entries in her personal archive.

Database Connectivity A lot of biological information is available in structured or semi-structured databases. Even though a lot of information is available, most databases contain information on protein interactions, and very little on reaction parameters.

The tool should disclose information in these databases to the user. This allows her to quickly add new reactants to the model that have an interaction with reactants already in the model. Since pathways consist of multiple reactants, adding a chain of reactions should work in the same fashion as adding a single reactant.

A first implementation could use only the directly referenced reactions from a single database, allowing the user to effectively cascade through the required reactions by hand. A display of reactions, aggregated from multiple databases, in which the user selects the reactants and reactions to import would be ideal.

The subject of aggregating and compounding information from multiple databases lies outside the scope of this research. An issue of note is the lack of standardised nomenclature for reactants, an example of this is the protein ELF3 also known as ESE1, ESX, ERT, or JEN [11, p. 208]. This issue can be solved in two ways: the nomenclature is standardised, or the tool supports aliases. The former appears to happen after time (i.e., 20 years or so) while the subject matures and is the better of the two solutions. The latter will be necessary to allow working with data from current research, and will most likely be the easiest approach.

Initial parameter values Two sets of parameters are available in the model: reactant parameters and reaction parameters. Both are required for the model to be executable.

The most influential reactant parameter is the concentration of reactant at $t = 0$. The tool should default the reactant parameters to predefined values based on common assumptions, and allow the user the change those assumptions based on personal preference.

Very little is known about reaction parameters [5]. The previously mentioned biological databases contain only sporadic information on reaction parameters.

Initial reaction parameters are dependent on a large amount of factors and biological background knowledge. The tool should offer the option to set initial parameters based on common assumptions about the reaction type, and the tool should allow the user to customise the assumptions to better fit her field of study or personal preference.

2.1.3 Refinement

Existing models are refined to offer a better match to reality. This scenario puts the user in the position of having to refine an existing model. During refinement a lot of small changes are made, based on the discovery of new information, either from literature or from empirical data, or just by way of trying out.

Refinement does not include the addition of new reactants or reactions. These issues are already covered in 2.1.2.

Comparison to empirical data The first step of refining a model with empirical data is the comparison of model predictions with the empirical data.

The empirical data must be made available within the tool to allow for computer-assisted methods of comparison. Therefore the tool should allow the user to import empirical data, and assign the loaded data to any value predicted by the model. At the time of writing the only predicted value is the concentration of activated reactant.

The actual comparison can be made visual by overlaying the graphs of the model prediction and the empirical data. The tool could emphasise the areas of the graph where prediction and measurement diverge outside of tolerance values.

Tolerance values are dependent upon multiple factors, one of which is the expected error in measurements. For the purposes of assisting the user in identifying discrepancies a customisable tolerance function could be introduced. Exact determination of functions usable in this context is outside the scope of the current research.

Most empirical data will contain information about multiple predicted values, and a graph with too much information quickly becomes difficult to read. Instead of presenting a confusing graph, the tool should assist the user by identifying the model elements participating in the discrepancy. A visual cue in the main display of the model will provide such assistance. Such a visual cue can be an emphasised and coloured border on a reactant, or an emphasised and coloured reaction.

Further work could be done to determine the viability of using biological common sense to assist the user in another way. For example, biological knowledge might be used to determine influencing reactions and reactants upstream that participate in the discrepancy even though they themselves are not measured.

Parameter fitting Once data is made available and discrepancies have been identified, the user will wish to change the model based upon this new information.

Although changing parameters by hand until a fit is reached probably improves the user's insight in the modelled processes, this is time-consuming work. The tool can assist the user by allowing her to declare some parameters as free parameters, and having the tool run multiple simulations.

Multiple strategies for the fitting should be offered. More on parameter fitting can be found in [5].

Advising on experiments The tool could gain a lot of utility by assisting the user in defining experiments to run in the lab.

This is a complex feature that requires at least the entering of biological queries and support for multiple versions of the same model. Although the subject of advising on experiments should be detailed in further work, the author has some observations.

Ideally the assistance of the tool boils down to the following: the user enters a hypothesis, and the tool will then provide a set of possible experiments that together will verify or falsify the hypothesis.

Each added reactant could be a hypothesis in itself (i.e., "Does this reactant take part in the pathway as modelled here?"), but other hypotheses might require more data from the user. To provide uniformity in interaction the hypothesis should be entered in the same way that biological queries (see 2.1.6)

are posited.

The tool should provide some way for the user to give preference to some types of experiments. This stems from the fact that some methods of measurement are more expensive then others (exactly which resources are used is immaterial, but examples include time, money and lab space).

Unless the advisory feature is taken very far, care should be taken to inform the user that the tool does not offer experiments that take reactants and reactions outside the model into account.

2.1.4 Collaboration in Meetings

The third scenario, Collaboration in Meetings, sees multiple users in the same room discuss and deliberate about the model.

During this discussion, changes to the model will be proposed and acted upon. The tool is available on a normal computer or laptop.

Attribution During the meeting multiple users suggest changes to the model and some of the changes will be decided upon by the whole group.

For later reference it will be necessary to trace back changes to the people that suggested them. Attributing each model modification to a single person (whether the modification was deliberated upon by multiple people or not) allows the tracking of responsibility.

This attribution is useful when explanation about the change is desired, or when the change has not been justified (for example, according to a reviewer).

Large visuals Two or three people can comfortably sit at a single computer screen, more viewers will become uncomfortable.

In a meeting it will be necessary for multiple participants to have a good overview of the model and any information relevant to the meeting. To this end, the tool should support a display mode for groups.

This group mode should support at least a beamer, to allow projection of the model onto a larger surface. This display mode should allow the users to set up the model display in such a way that information relevant to the meeting is emphasised while other information is still available.

The group collaboration might take advantage of previous and current work in the direction of collaboration and alternative displays such as touch tables [15].

2.1.5 Review

Models, just like papers and data sets, should be reviewed. This scenario puts a user to the task of reviewing a model.

Such a review will have to take into account that a model contains a large amount of assumptions and guesses.

Record of evolution and creation Seeing how a model was created provides insight into the choices made during creation. To assist the user with this, the tool should keep a record of the actions taken to create the current version of the model.

Each change should be accompanied by some information about the change: the author of the change, and a short note about the change.

When reviewing the reviewer should be able to view previous versions of the model, and see how the current version evolved from it. This feature ties in with the Confidence and Multiple Versions features (both found in 2.1.6).

Justification of model elements When modelling, the user has to make choices: which assumptions to use, what reactants to add, the parameters of a reaction, and so forth. These choices have to be recorded; this is usually done in a separate document.

The model has to be reviewed together with these choices, since the choices are part of the modelling process. The model can only be fully reviewed of this parallel document is available. To ease the review of this parallel document, the tool should allow the author of a model to include the justification of choices she made in the model itself.

By integrating the justification the author of the model is forced to think about the justification of her choices and make them explicit. Section 2.3 goes into more detail about the annotations themselves.

Having justification integrated with the model will also aid the publication of models, since a list of literature references and assumptions can be generated directly from the model, mitigating the problem of out-of-synch parallel documents.

The tool should offer at least an option to check whether all model elements are annotated with a justification. This feature will not be able to determine the validity of the available justification, but it can warn the author about lacking justification.

2.1.6 Exploration

The exploration scenario has the user playing around with the model. By changing the model, and seeing how the changes affect the model's execution a user can explore the different facets of the model.

Exploration of the model and its dynamics increase the insight of the user. Therefore, playing with the model should be supported in an intuitive way, as the user's exploration should not be hindered by clumsy interaction.

Confidence The user must have confidence that the tool will not allow her work to be lost. To this end, it must be clear to the user that all her changes can be undone, to retrieve the starting model. The tool should allow the user to undo any changes, and in this way retrieve previous versions of the model.

Preferably the changes should be stored with the model. This allows her to send the model somewhere else, or continue working on it another day and still go back to old versions.

Small changes are best handled by the familiar undo/redo functionality available in many programs. Large changes are best handled as revisions in a version management system; this allows the user to add some note justifying the change.

Previous work has already been done on the handling of revisions of data, though these focused on handling plain text [24, 22, 7, 14, 19]. Concepts and methods from these version control systems still apply to the versioning of the model. Notes for further work can be found in section 6.3.

Multiple versions When trying out new things it is useful to quickly look back at other versions of the model. Having multiple versions of the model allows the user to compare them and see how they react differently. The tool should support multiple versions of the same model to support the user in her exploration of the model.

Model execution data could be compared to other versions in much the same way that empirical data is compared to non-empirical data (see 2.1.3).

Multiple versions of the model also allows collaborators to quickly branch of the 'main' version of the model to create a variant for demonstration or testing. These branches can later be reviewed in the same way as the original model, and multiple variants of the model may support differing hypothesis.

Biological queries Having the ability to change the model and see the results of simulating the new model gives the user insight into the model's behaviour.

Trying to reach a certain goal is difficult this way, since a complex model might well show unexpected behaviour. The option of querying the model for the occurrence of certain states allows the user to focus on interesting behaviour.

Determining how the user inputs and executes biological queries is beyond the scope of this work. [17] offers a good start for further work on this subject.

2.1.7 Integration

Models on the same subject but by different people will vary. The same goes for models from different projects created by the same person. This scenario puts a user in the situation in which she wishes to integrate two models.

The integration of models allows users to join their work, and create a more complete model. This might offer new insights as the models need to be reconciled and changed to fit.

Recognition of matching network elements By assisting the user in the recognition of matching model elements the tool helps to speed up the laborious parts of integration, allowing the user to focus on the actual problems of the integration.

The tool should allow the user to integrate by either importing (part of) a model from another file, or by pasting (part of) a model into the current model.

After the addition of the model fragments to be integrated the user makes a selection of those elements she wishes to match automatically. The tool can then try to find matching model elements. This matching is useful at the structural level, so reactions and reactants need to be matched, while the actual parameters should be reconciled afterwards.

The visual layout of a network is irrelevant to the semantics, but the structure can be used to find common components. The common component search could be aided by using the reactant name as an indicator of equality, this will increase the number of correct matches. As already noted in 2.1.2, care must be taken as a single reactant can have multiple names depending on the author(s) of the model.

After the tool has determined matching model elements it should offer these matches to the user so she can determine which matching she wishes to use, and which to discard. The tool will then merge the accepted matches, and defer to the user on the method of reconciling the parameters of the matched elements.

The exact biological implications of applying such a merging strategy for the model element parameters still need investigated.

A first implementation might use only exact naming or alias match, and offer the choice of using one of the unchanged parameters, or an average of both.

Record of integration Multiple models will most likely mean multiple authors. It is necessary to keep track of the origin of integrated model elements to allow reviewers and collaborators to quickly determine whom to contact should they wish to discuss the model.

If possible, the creation history of the integrated model should be added to the history of the current model, since this allows reviewers to see the development of both parts of the integrated model.

This feature should be considered together with the Record of evolution and creation (see 2.1.5 and 2.1.6).

2.2 Cytoscape Integration

The analysis of the usage scenario's points out a lot of possible features for the tool. However, before these features can be implemented, the tool should be integrated into a more robust framework.

The primary users of the tool are biologists, so integrating the tool in a well-known biology workbench allows them to start working with the tool in a familiar environment.

This section of the chapter describes the choice of visualisation workbench, and the design of the integration with this workbench.

2.2.1 Choice of Visualisation

A well-known and commonly used biology workbench is Cytoscape [21]. Cytoscape is a network visualisation workbench providing only visualisation and generic network editing capabilities. Although other visualisation workbenches exist, such as PathVisio [26], Cytoscape has been chosen because it appears to be the de facto standard.

Cytoscape's visualisation framework is flexible enough to change almost every aspect of the visualisation based on the parameters of the network elements (i.e., nodes and vertices).

Since there are hundreds of plugins available for Cytoscape there is a lot of documentation, albeit in the form of examples. One example is Cerebral [3]. Cerebral is mentioned specifically because it has functionality for displaying data that change over time and functionality for laying out networks while taking into account the location of the network's components in a cell. The visualisation facet of Cerebral comes close to what our tool should offer by way of visualistion.

2.2.2 Two-tier model generation

The formal model used by UPPAAL needs to be generated from the network in Cytoscape. However, the Cytoscape network does not contain the required information in a readily useable format, so a conversion is required.

This conversion is designed as a two-tier process. The Cytoscape model is first converted into an intermediate model, and this intermediate model is then converted into an UPPAAL model. This two-tiered approach adds flexibility to the conversion process. Among the advantages of the two-tiered approach is the possibility to replace either the Cytoscape front end, or the UPPAAL back end without having to rewrite large parts of code.

Another advantage is the ability to create other tools, such as analysers or possibly a summary generator, that use the same intermediate format. These tools should not have to handle the Cytoscape network, which is not specific enough, or the UPPAAL model, which is too specific.

2.2.3 Time Slider

The previous version of the tool, INAT (see 1.1), offered time-based display through snapshots. Such a display of the changes over time offer less insight into the dynamics of the model, as the user is constantly switching back and forth between two static images.

To allow the user to view the dynamics of the model herself the tool should offer her the option of seeing the dynamics of the model unfold. This is facilitated with the time slider.

The time slider represents the complete span of time of the simulation. Whenever the user positions the cursor of the time slider, the visualised model is updated to reflect the reactant activation at the indicated time point. This

direct visual feedback allows the user to move back and forth in time to see the interaction of the different reactants.

This interaction is familiar to the user because most multimedia programs use the same form of interaction to allow the user to slide to a different point in time in a video or audio stream. Due to time constraints the familiar play/pause, rewind and stop buttons have been left out. These could be added to heighten the users' familiarity and provide a more direct way of navigating the time span.

Notes on the implementation of the Time Slider can be found in 3.3.

2.3 Annotations

This last section of the chapter presents the design of annotations. As described in section 2.1.5 annotations are used to justify model elements.

The annotations in a model are used as a lab journal. They describe why model elements exist, and how their parameters were determined.

Annotations are also useable as a method of communicating with collaborators. They allow a user to communicate her intent and reasoning about the annotated model element.

All model elements, including the parameters of reactions and reactants, are candidates for annotation. Each model element can have multiple annotations, but annotations are not shared among model elements. The implications of this will be discussed in 4.2.

2.3.1 Dimensions

The annotation dimensions presented here are inspired in part by the annotation scheme in [23].

Each annotation consist of four fields: Source, Type, Reference and Statement & Note. A fifth field, Confidence, was considered, but has been left out (see 4.3 for a short discussion on this). The addition of a confidence feature is a good direction for further work (see chapter 6).

Source The source of the knowledge referenced with this annotation. The source dimensions are as follows:

- Literature: the knowledge is attributed to a paper, book, or other form of peer-reviewed work.
- Database: the knowledge is attributed to a database of biological knowledge.
- Other: the knowledge is attributed to something else, such as a person or the current study.

Most databases contain justifications such as references to previous work. The inclusion of the 'database' source allows the user to indicate this.

The 'other' type is included to allow the user to refer to the current study, a colleague's conjecture, or some other source of information not included in the given dimensions.

Type The type dimension indicates the type of statement this annotation references. This dimension indicates the type of statement as the author of the statement intended it. The type dimensions are defined as follows:

- Conjecture: the statement is based on conjecture,
- Hypothesis: the statement is a hypothesis,
- Measurement: the statement is a measurement,
- Asumption: the statement is an assumption,
- Fact: the statement is well-established knowledge.

The type dimension aids the user in determining the weight of the knowledge. A justification based on a fact is a stronger justification than one based on conjecture. Due to time constraints, a dimension for the subjective judgement of the annotator has been left out of this design. More on this can be found in section 4.3.

Reference The reference dimension is the actual reference to the knowledge. The reference can be given in any form, but will be most useful when given in a recognisable and consistent form.

Structured input of the reference would be more useful for exporting of references. However, most users seldom appreciate the lack of freedom when entering data that can take on many different formats.

Statement & Note The statement and note dimension is used to indicate how the knowledge is used in the model.

A summarised statement allows the user, and any other users, to quickly glean the knowledge referenced in this annotation. The statement field can also be used to note how the referenced knowledge is used in the model.

Chapter 3

Implementation & Result

The integration with Cytoscape is aimed at providing an extensible framework for later work. All work presented in this chapter was done by the author, unless mentioned otherwise. Where possible the implementation takes advantage of facilities provided by Cytoscape. The integration with Cytoscape consists of three major components (figure 3.1 offers an overview of these components):

1. The intermediate model, described in detail in 3.1, which connects the Cytoscape model with the UPPAAL model (displayed in the top half of figure 3.1).
2. The UPPAAL invocation and interpretation of the results, described in 3.2 (seen in figure 3.1 in the bottom-right corner).
3. The time slider control to allow users to view different time points in the simulation (part of the 'Plugin' element in in figure 3.1).

Next to the basic facilities of Cytoscape (e.g., saving and loading of models, simple model editing operations), one advanced facility of Cytoscape is heavily used by the integration: the visual style mapper also known as the VizMapper. The VizMapper determines the visual styling of nodes and edges based on the attributes of those elements.

To allow the largest amount of freedom to users, no programmatic changes are made to the visual styling. This offers users the greatest amount of freedom when determining how to display their model.

The presented implementation can be found in a Subversion repository located at `http://hmisvn.ewi.utwente.nl/ExBio/projects/inat/`. At the time of writing this repository does not allow public access.

3.1 Intermediate model

The intermediate model is the 'in between' translation from Cytoscape to UPPAAL. The model offers a simple data structure to allow easy operations on the

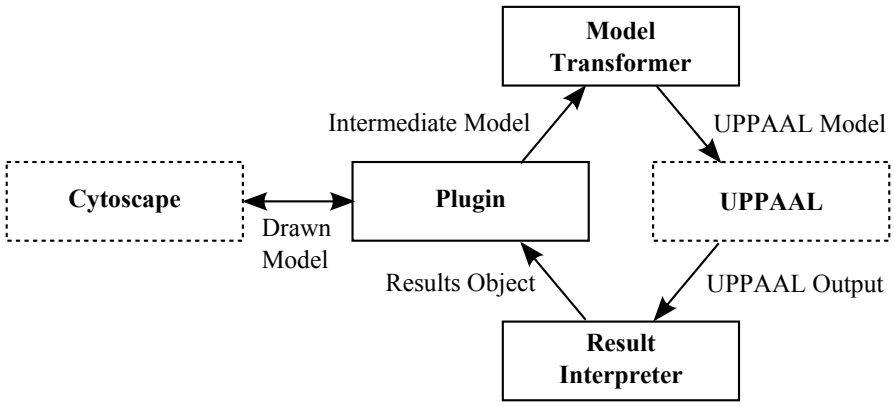

Figure 3.1: Cytoscape integration components and the flow of information during model simulation.

model (see 2.2.2).

3.1.1 Structure

The intermediate model is structured as a set of model entities (reactants and reactions) contained in a model object. It is very simple. A model entity is any element of the model that can exist independently of other model elements. Each model entity has a unique identifier. At the time of writing reactants and reactions are the only existing model entities.

The model object itself and every entity in the model has a set of properties. The set of allowed properties is not constrained, and the meaning of a property's value is left to the clients of the model object.

To ease the use of the intermediate model implementation each property is implemented as a simple object. The properties are implemented as distinct objects to provide type-safety at runtime. Thus, each property is defined by a $(key, type, value)$ pair. At the time of writing the allowed types are limited to: strings, booleans, integers, floats and two-dimensional arrays of integers (represented by a custom table type).

The type-safety of the model is implemented by using assertions to execute the type checks. This is viable since most of the type errors will occur during development of the tool. Once a stable and tested version has been created the assertions can be disabled.

3.1.2 Relation to Cytoscape model

To enable other tools to process the model created in Cytoscape (the 'Drawn Model' in figure 3.1) it has to be converted to the intermediate model format. To convert the Cytoscape model to the intermediate model, the following steps are taken:

1. All model parameters are checked for validity. This is important for parameters like the maximum number of activation levels. If any of the checked values is unset, or does not correspond to the correct value, an exception will be thrown.

2. An intermediate model object is created and all model-wide properties are set on the intermediate model.

3. Each node in the Cytoscape model corresponds to a single reactant and is converted by creating the reactant entity, setting the required properties on it (i.e., the name and initial concentration) and adding it to the intermediate model.

4. Each edge in the Cytoscape model corresponds to a single reaction. It is converted by looking up the reactant identifiers (one for self-reactions, two for reactions involving two reactants), creating the reaction entity, setting the required properties on it (i.e., the reaction type, the identifiers of the involved reactants, the reaction timing table and whether the reaction is activating or inhibiting), and finally adding the reaction to the intermediate model.

During the last step of the conversion the reaction timing tables are filled based on the information in the Cytoscape model.

3.1.3 Relation to UPPAAL model

Once the Cytoscape model has been converted into an intermediate model, the intermediate model is converted into an UPPAAL model by application of the model transformer (the 'Model Transformer' in figure 3.1).

Since the conversion into an UPPAAL model is not trivial, this step is abstracted by using a transformer interface. This interface allows one to quickly change transformer implementations. UPPAAL models are written in a mix of XML structured data and character data for declarations and definitions. More on this can be found in [4, appendix B].

The basic model transformer implements the conversion to the variables model. The variables model was designed by Stefano Schivo. This model uses variables to store the reactant activation levels instead of relying on the model's structure to allow activation level manipulation. The following steps are taken when the model is transformed with this transformer:

1. The preamble of the XML document is written. This includes the correct encoding (utf-8), the doctype declaration, the opening of the declarations element, and the declaration of a global clock.

2. The requisite constants, channels and variables are written. This includes channels for each reaction in the intermediate model, and variables for each reactant.

3. The declarations element is closed, and all UPPAAL templates that will be used in the composition of the final system are written.

4. The template instantiations for both reactants and reactions are written, as well as the process coordinator instantiation.

5. Finally the final system composition containing all instantiated processes is written and the XML document is closed.

A number of utility functions handle the conversion of time values, and the determination of reaction identifiers. The time conversion takes into account the fact that a constant signifying the 'infinity time' is embedded in the templates. Reaction identifiers are based on the involved reactants and whether the reaction is inhibiting or activating.

3.2 UPPAAL invocation

After the intermediate model is converted into an UPPAAL model, it is handed over to UPPAAL for further analysis. The UPPAAL invocation itself is exactly as described in [4, appendix A.2], and generates output according to appendix C.2 of the same.

The UPPAAL output is analysed by a result interpreter (the 'Result Interpreter' in figure 3.1). The result interpreter is an abstract interface that accepts the intermediate model and the UPPAAL output and produces a results object. Currently the only result interpreter available accepts results as produced by the UPPAAL invocation and interprets these as levels of activation over time.

The only implemented results parser is one that creates an activation level results object, this object is described in 3.2.1. The UPPAAL output is parsed by setting up a temporary data structure for bookkeeping and analysing each line of the UPPAAL output as follows:

1. Any line that does not, by itself, define a state is ignored.

2. The first line of the UPPAAL output that constitutes a state is taken as the initial state. All reactants found in this line are assumed to participate in the rest of the results and are added to the temporary data structure.

3. All following lines that constitute a state are accepted. A regular expression is used to extract the time information from the line. Once the time indication has been retrieved, the line is searched for reactant activation levels and the intermediate results structure is updated.

After all lines have been analysed, the activation level results object is created from the bookkeeping data. If at any time a line is found that constitutes a state, but does not indicate a global time value an exception is thrown.

3.2.1 Activation level results object

The result parser returns a result object (seen with the same name in figure 3.1). This object contains information on the activation levels of all reactants in the model. The results object is can be queried for the activation level of a single reactant at any point in time within the span of the simulation.

Internally the results object represents this information by only storing the level of activation of a reactant at the point in time at which it changed to that level. This compresses the amount of required memory, especially for simulations over a long time span while still allowing quick interrogation of the results object.

3.3 Time slider

As described in 2.2.3, the time slider is the user interface control allowing users to slide through the time window of the simulation results.

The implementation of the time slider leverages the flexibility of Cytoscape's VizMapper. When the user manipulates the time slider, the node attribute of each node corresponding to a reactant available in the activation level results object is updated with the activation level at the indicated point in time.

By changing a node attribute in the Cytoscape model to reflect the concentration of the reactant at the point in time indicated by the slider, the actual method of displaying the time-based concentration information is dictated by the visual styling set by the user.

Chapter 4

Discussion

4.1 Cytoscape Integration

Cytoscape has been designed as a workbench to view and modify graph-based networks, and to let plugins handle analysis and other algorithms operate on these networks. The Cytoscape plugin structure was created with the idea that plugins should concentrate on modifying or augmenting the network data. Due to this clear focus on communication through the data, Cytoscape lacks the flexibility to really change the way the user interacts with the model.

The Cytoscape integration is implemented as a plugin, and not as a modification of Cytoscape itself to stay compatible with other Cytoscape plugins. However, during implementation it became clear that the current version of Cytoscape does not allow a lot of freedom in modifying the user interaction with the workbench.

Since the Cytoscape integration of the tool requires multiple interaction changes, it will be difficult to continue implementation in the current version of Cytoscape. A new version of Cytoscape is in the making, and this version may support the modification of user interaction on a higher level, but this is not certain. To obtain complete control over the user interaction with the tool it might be necessary to steer away from Cytoscape, and offer a specialised user interface.

Remote simulation The current method of integration with Cytoscape has the benefit of providing an exchangeable backend. This can be used to exchange the current backend with a backend that uses a remote machine to execute the actual simulation.

Such a remote backend allows for quicker installation of the plugin, since none of the normal UPPAAL software needs to be installed on the user's workstation. When used outside of a single institution this kind of remote execution brings with it all kinds of legal and security issues, so this subject should be approached with care and consideration.

While the concept of remote simulation is intriguing, the legal and technical pitfalls could prove to be out of proportion with respect to the utility the tool gains from further development in this direction. It is, of course, possible to mitigate legal and security issues by implementing the remote not as a service, but as a dedicated server that needs to be set up by the end-user (or their technical department).

Progress Indication During the execution of the simulation the user is presented with a dialogue containing a progress bar. The Cytoscape plugin structure intends for this progress bar to show the progress of the current task. To show the amount of progress one would need to know the total of amount of work, and the progress over this amount. Thus the determination of progress should be possible as the current implementation only uses a reachability query, which boils down to querying whether a certain point in time can be reached.

However, due to the way UPPAAL executes the simulation no information on the current level of progress is available until the simulation has finished, effectively disabling the use of the progress bar as intended by Cytoscape. This means that the user has no indication of the progress of the simulation.

4.2 Duplicate Annotations

The current annotation design does not take into account that annotations might have to be duplicated among multiple model elements. This duplication could have been prevented by allowing a model element to reference previously added annotations. Allowing a single annotation to be referenced by multiple model elements would also require more complex annotation management, something that might deter from the user's desire to use annotations.

However, the current design encourages the user to add duplicate annotations. Having duplicate annotations will result in redundency of information. This not only increases the amount of work that needs to be done manually, but having duplicate annotations makes the automation of annotation tasks (such as exporting a list of assumptions, or the generation of a list of referenced literature) more complex.

It is clear that the current situation, which requires the user to create duplicate annotations, is not desirable. Determining how to solve this problem, and how much of the annotation management might be automated is something for the future.

4.3 Annotation Confidence

During the design of the annotations (see section 2.3) the issue of confidence came up. One of the main goals of the tool is to enable collaboration; the issue of annotation confidence should not be disregarded. However, confidence has been left out of the current annotation design due to time constraints.

Unlike the other dimensions, confidence is greatly dependent upon the annotator. Since the level of confidence the annotator has in any single annotation is highly subjective, any form of confidence should take into account the possibility of multiple opinions by different collaborators. Even if the confidence dimension were structured to contain a single scalar confidence per collaborator, there is no clear-cut way to reconcile these different opinions into a single confidence verdict for the annotation. Section 6.6 indicates directions for further work.

4.4 User Guidance

Section 2.1.5 highlights the importance of justifying the choices that were made during the creation and modification of the model. The annotations take on part of the role of a lab journal in an actual laboratory because they also allow the user to record notes and add clarifications to the model while it is being created. Because of this importance the interaction with the tool should encourage the user to add annotations to the model.

The user will be annoyed if the tool forces her to add an annotation for each change she makes in the model. But such interactions as adding a new reactant or reaction should certainly provoke the user into adding a justification to the newly created model element. Unless the new version of Cytoscape allows plugins to modify the way networks are created and altered this kind of interaction might not be feasible in Cytoscape.

4.5 Limitations on User Interaction

A last point of discussion is the dependency of user interaction upon the chosen data model, and the dependency of the data model upon the user interaction. This dependency is distinct from the used visualisation framework or execution backend, and holds regardless of the chosen visualisation or backend.

The current tool was created for the visualisation and analysis of kinase networks. This implies a certain way of displaying information: the diagrams drawn in the tool do have a mapping to the actual chemical formulae, but this mapping is not trivial. Should the user wish to model a different kind of chemical network, this will require changes to the data model.

As the data model is changed, this might change the way the network is displayed, which in turn changes the user's perception of what is 'upstream'. This change in perception of the user requires that interaction based on the notion of 'upstream' model elements is revised to correspond to the changed model. Because user interaction is the glue between the data model and the user, it is dependent upon the data model and the user's perception of this model.

This dependency works both ways. If, for example, the user wants to interact with the tool in a way that does not require her to add duplicate annotations, this will imply a change in the data model to accommodate this changed in-

teraction. This will of course provoke changes in other interactions as the data model is changed to accommodate. In this way the data model is dependent upon the user interactions, because the data model itself would be meaningless to the user if her interaction with the tool would not result in a perceptible change in the data.

With both dependencies in place we can conclude that further development on either the tool's user interaction or the tool's data model will require attention to both.

Chapter 5

Conclusions

Through analysis of the tool's probable usage scenario's I have derived a list of features that would increase the usefulness of the tool. The derived features offer a good way to determine which directions further work could explore. The following features are of special note because they seem to be good direction for immediate further work and implementation:

- Pathway imports and database connectivity (2.1.2),
- Comparison to empirical data (2.1.3),
- Record of evolution and creation and multiple versions (2.1.5 and 2.1.6),
- Justification of model elements (2.1.5) and the annotation structure (2.3),
- Selection strategy and group operations (2.1.1).

Secondly, I have integrated the tool with Cytoscape. This integration opens up a larger audience to the tool, and allows the use of an impressive array of Cytoscape plugins. The display of the simulation results (i.e., the time-dependent activation level) of the old tool has been replaced with the time slider. The time slider has been received with enthusiasm at the International Symposium on Integrative Bioinformatics 2011 (IB2011) in Wageningen.

And finally, an annotation structure has been designed that allows users to justify elements of the model without using an external document. While this annotation structure has not yet been implemented, a first implementation should be straight forward.

Chapter 6

Further Work

6.1 Features

The features mentioned in chapter 2 are not implemented, and most will require further research before they can be implemented in full. Although all of these features have been presented in the chapter 2, a short overview is given here categorised by usage scenario.

Miscellaneous Features that are important in all scenario's.

- Visualisation of information (2.1.1). The tool contains a large amount of information. This information needs to be visualised for consumption by the user, preferably in a user-customisable way.
- Selection strategy (2.1.1). Implementing different selection strategies allow the user to expand or contract selections based on the biological meaning of the elements in the model.
- Group operations (2.1.1). By making certain operations capable of affecting a group of model elements the user is assisted in making large modifications to the model.
- Copy & Paste (2.1.1). By allowing familiar copy and paste operations without losing associated data or meta-data the user is assisted in the creation of repeated patterns.

Creation Creating a new model from scratch.

- Published pathway importing (2.1.2). Allowing the use of a pathway repository, either remote or local, and the import of SBML files.
- Database Connectivity (2.1.2). Retrieving relevant data from biological databases. This feature focuses on the retrieval of small parts of a pathway, either with or without downstream reactants.

- Initial parameter values (2.1.2). Determining the initial values for the reaction parameters is a complex issue requiring background knowledge on the biological relevance of these reactions.

Refinement Adapting an existing model to fit new information.

- Comparison to empirical data (2.1.3). The comparison of simulation data with empirical data can be made accessible in a number of ways.
- Parameter fitting (2.1.3). Fitting reaction and reactant parameters to the measured data by hand is time-consuming.
- Advising on experiments (2.1.3). By using biological knowledge the tool can suggest experiments to test a hypothesis posed by the user.

Collaboration in Meetings Multiple users discuss and modify a model.

- Attribution (2.1.4). Changes and additions during a collaborative meeting should be attributed to the user that suggested them.
- Large Visuals (2.1.4). The visual representation of the model during collaborative meetings should be adapted to beamers or other media used during such meetings.

Review Peer-review or progress review of a model.

- Record of evolution and creation (2.1.5). During the creation and alteration of the model a record of the steps taken by the user should be created to allow for reviewing not only the model itself, but its creation as well.
- Justification of model elements (2.1.5). By adding an annotation to each model element, the user can justify the choices she made during the creation of the model.

Exploration Playing around with the model to see how it behaves.

- Confidence (2.1.6). The tool should behave in a predictable way and offer a robust system to make sure the user does not lose any work while playing around. This includes, but is not limited to, implementing a undo/redo system.
- Multiple versions (2.1.6). Supporting multiple version of the same model allows the user to quickly try out different things without fear of losing a version, or the hassle of manual file management.
- Biological queries (2.1.6). Allowing the user to pose questions about the model's behaviour enables her to explore the model in different ways.

Integration Combining multiple models into one.

- Recognition of matching network elements (2.1.7). By recognising which model elements in a selection actually refer to the same thing in biology will allow for assisted integration when combining two or more models from different sources.
- Record of integration (2.1.7). Keeping a record of where model elements came from allows the user (and her reviewer) to identify the source of model elements even after further modifications.

The selection strategy and group operations feature (2.1.1) and the record of evolution and creation and multiple versions (2.1.5 and 2.1.6) features are good candidates for immediate implementation. This is also true of the implementation of the visual cues for comparison of empirical data as described in section 2.1.3.

6.2 Selection strategy and group operations

Selection strategies are described in 2.1.1. Four strategies have been outlined there that will prove easy to implement.

The current version of Cytoscape allows group operations in a fashion. The data display pane will show all data of the selected model elements, but this information is in an unfriendly format.

Both the selection strategy and the group operations have a straight-forward implementation in the current version of the tool. This will allow for a quick increase in usability. However, further research could be done to hone these features to the point of becoming natural operations for the user.

6.3 Record of evolution and creation

Much work has been done on the subject of versioning. Well known versioning systems include CVS [22], Subversion [7], Mercurial [19] and Git [14]. Of these systems, CVS and Subversion have a server-client model, which adds technical dependencies. Mercurial and Git are based on a distributed model.

When working on this feature, one should keep in mind the difference between model semantics and file semantics. It is important that the chosen versioning system understands the relevance of different model objects.

6.4 Specialised user interface

Should the path of a specialised user interface be chosen (as discussed in section 4.1), this interface will need to fulfil certain requirements. Determining these requirements and implementing the framework of the new user interface will require further research and design.

Care should be taken to ensure that the new user interface allows quick modifications of most graphical parts while maintaining a strict and well-defined structure.

6.5 Publishing annotations

The current format of annotations (2.3) allows for much freedom of input. Should the reference dimension of the annotations be set in a more structured format, this will pave the way for tooling that allows the extraction of annotation information.

Such tooling can be used to publish or extract a list of referenced literature as an appendix to the model. Further work should be done on this facet of the annotations and the uses that can be made of having more structured reference information in the model.

6.6 Annotation confidence

Section 4.3 discusses the subject of annotation confidence. The current design offers the Statement & Note dimension, in which a rudimentary confidence level can be recorded.

A simple solution would be to add the Confidence dimension to the annotations. This dimension is structured as a list of (collaborator,confidence level) pairs and records the confidence verdict of each collaborator.

Further work should be done on this to determine whether this kind of confidence measure is adequate, and how this should be presented to the user.

6.7 User Studies

To tune the user interaction with the tool's multitude of features we need more information on the workflow used by users of the tool. The same is true for the prioritisation of possible features and their exact implementation.

Further work on the development of features should at least include a regular cycle of implementation and feedback. Those features that still require further research will benefit greatly from a user study to determine how users interact with the tool.

Bibliography

[1] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Dennis Bray, Karen Hopkin, Keith Roberts, and Peter Walter. *Essential Cell Biology, Second Edition*. Garland Science/Taylor & Francis Group, September 2003.

[2] R. Alur and D. Dill. Automata for modeling real-time systems. *Automata, languages and programming*, pages 322–335, 1990.

[3] A. Barsky, T. Munzner, J. Gardy, and R. Kincaid. Cerebral: Visualizing multiple experimental conditions on a graph with biological context. — *IEEE Transactions on Visualization and Computer Graphics*, pages 1253–1260, 2008.

[4] W.J. Bos. Interactive signaling network analysis tool. Master's thesis, University of Twente, the Netherlands, Enschede, 2009.

[5] W.W. Chen, M. Niepel, and P.K. Sorger. Classic and contemporary approaches to modeling biochemical reactions. *Genes & development*, 24(17):1861, 2010.

[6] Federica Ciocchetta and Jane Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410:3065–3084, August 2009.

[7] B. Collins-Sussman, B.W. Fitzpatrick, and C.M. Pilato. *Version control with subversion*. O'Reilly Media, Inc., 2004. Available online: `http://svnbook.red-bean.com/`.

[8] Hidde de Jong, Johannes Geiselmann, Céline Hernandez, and Michel Page. Genetic Network Analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–344, February 2003.

[9] Lorenzo Dematté, Corrado Priami, and Alessandro Romanel. Modelling and simulation of biological processes in BlenX. *SIGMETRICS Perform. Eval. Rev.*, 35:32–39, March 2008.

[10] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[11] MB Goldring, M. Otero, K. Tsuchimochi, K. Ijiri, and Y. Li. Defining the roles of inflammatory and anabolic cytokines in cartilage metabolism. *Annals of the rheumatic diseases*, 67(Suppl 3):iii75, 2008.

[12] David Harel, Yaki Setty, Sol Efroni, Naamah Swerdlin, and Irun R. Cohen. Concurrency in biological modeling: Behavior, execution and visualization. *Electron. Notes Theor. Comput. Sci.*, 194:119–131, January 2008.

[13] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):134–152, 1997.

[14] J. Loeliger. *Version control with Git*. O'Reilly Media, 2009.

[15] J. Logtenberg. Multi-user interaction with molecular visualizations on a multi-touch table. Master's thesis, University of Twente, the Netherlands, Enschede, 2009.

[16] Pedro Mendes, Stefan Hoops, Sven Sahle, Ralph Gauges, Joseph Dada, and Ursula Kummer. Computational modeling of biochemical networks using COPASI systems biology. volume 500 of *Methods in Molecular Biology*, chapter 2, pages 17–59. Humana Press, Totowa, NJ, 2009.

[17] P.T. Monteiro, D. Ropers, R. Mateescu, A.T. Freitas, and H. De Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24(16):i227, 2008.

[18] M. Nagasaki, A. Saito, E. Jeong, C. Li, K. Kojima, E. Ikeda, and S. Miyano. Cell illustrator 4.0: A computational platform for systems biology. *In Silico Biology*, 10(1):5–26, 2010.

[19] B. O'Sullivan. *Mercurial: the definitive guide*. Oreilly & Associates Inc, 2009.

[20] J.C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*, pages 61–71. IEEE, 2007.

[21] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498, 2003.

[22] D. Thomas, A. Hunt, and P. Programmers. *Pragmatic Version Control: Using CVS*. Pragmatic Bookshelf, 2004.

[23] Paul Thompson, Raheel Nawaz, John McNaught, and Sophia Ananiadou. Enriching a biomedical event corpus with meta-knowledge annotation. *BMC Bioinformatics*, 12(1):393, 2011.

[24] W.F. Tichy. Rcsa system for version control. *Software: Practice and Experience*, 15(7):637–654, 1985.

[25] M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. A. Hutchison. E-CELL: software environment for whole-cell simulation. *Bioinformatics*, 15(1):72–84, January 1999.

[26] M. van Iersel, T. Kelder, A. Pico, K. Hanspers, S. Coort, B. Conklin, and C. Evelo. Presenting and exploring biological pathways with PathVisio. *BMC Bioinformatics*, 9(1):399, 2008.

[27] M.Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the working conference on Advanced visual interfaces*, pages 110–119. ACM, 2000.