A Tool for Visualising Cell Model Results
MInf Project Phase 2

James Hulme
s0901522

January 18, 2014

Abstract

This is where the abstract will go

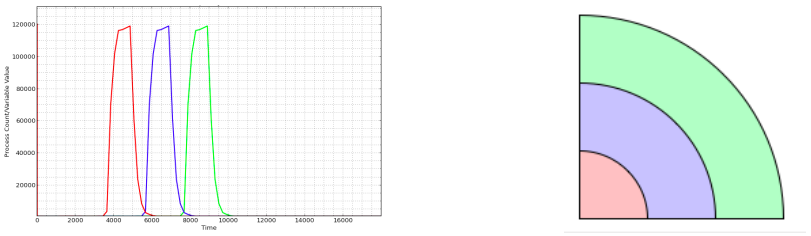
Contents

1 Introduction

Biologists often use computer models to help guide their research as modelling is much cheaper than experimentation. There are a number of tools available for biological modelling. These tools typically require a certain level of numerical confidence to create and interpret. Not all biologists have this numerical confidence. Some researchers find writing and interpreting models a challenge, this can make them less effective in their research. It is therefore necessary for these tools to help them relate the data to their field by incorporating domain knowledge.

One such tool that can be used for modelling is Bio-PEPA, an extension of the PEPA process algebra. Bio-PEPA is currently implemented as a plugin for the Eclipse IDE. Bio-PEPA visualises the model results as time-series graphs. There is one team, the Src researchers, in particular who use Bio-PEPA and do not have the numerical confidence, as described above, to be comfortable using Bio-PEPA. This team is the focus for this project. The purpose of this project was to extend Bio-PEPA's visualisation capabilities to allow the previously mentioned team, and other similar users of Bio-PEPA to more effectively analyse their results.

A significant problem with Bio-PEPA's visualisation capability is that it is difficult to represent spatial change on a time-series graph (as can be seen in Figure 1). In Bio-PEPA you can have a species at different locations in the cell, for example, next to the nucleus, next to the cell membrane and throughout the cytoplasm. The movement of this species through the cell can by modelled by seeing the population of it in each location over time. This is difficult to visualise on a time series plot as three lines is too abstract. It requires the use of biological metaphor to be easily interpreted. In this case a visualisation of the cell that can show how the species moves through the cell. It is this sort of inference that the Src researches find challenging to do with Bio-PEPA currently.



(a) *Time Series Representation* **(b)** *Spatial Representation*

Figure 1: *One species at three locations in the cell represented traditionally on a time series graph and also spatially in a cell*

Over the course of the project the scope has been expanded. The original objective was to assess which forms of visual representation are most helpful and informative to laboratory science. At the end of the first project phase the object changed (to reflect that it was more of an engineering project than an experimental project) to be to develop a tool to visualise the results of dynamic, time-series models of intra-cellular behaviour based on biochemical reactions. This objective was focused on visualisation to aid those researchers who are not numerically confident. The second phase of the project has added to this objective to also aid interpretation and collaboration. This change in objective is to make the tool better for all users.

1.1 Where Does This Tool Fit In?

In the first stage of the project a review was performed of the features of a number of modelling and visualisation tools. This review included specialised software aimed at biologists and general software for anybody doing data visualisation.

The software that was reviewed were: Bio-PEPA, Uppaal, V-Cell, Cell-O, Copasi, Cell Designer and WEAVE. Bio-PEPA, V-Cell, Cell-O, Copasi and Cell Designer have been written for biological modelling. Uppaal is modelling software for general use, and WEAVE is a general data visualisation tool.

All offered some level of visualisation, some simply graphs, others more complex visualisations. Bio-PEPA offered only line graphs. Uppaal had visualisations that highlighted where in a finite state machine view of the model the current state is. V-Cell had visualisation of the model in a hierarchical set of circles, it was also display spatial elements of the results data by displaying a heat model view of the cell. Cell-O was aimed more at multi-cell models and was able to show them moving and splitting, it also had visualisation of the model as a finite state machine. Copasi only graphs although the user had more control over the display of the plots. WEAVE had the largest visualisation capacity being able to display a variety of standard graph times along with more interesting ones, such as geographical maps, but it did not appear to have anything specialised for biological models. WEAVE is also the tool that gave the user the most control over the visualisation.

The existing biological modelling software seems to be focused more on the ease of modelling. The visualisation features on offer are typically quite basic. They also lack the more innovative features that can be found in the newer general data visualisation software.

The project scope expanded to include goals not specifically related to visualisation. A software review was performed of software the allowed for collaborative editing to see what features are common place, which are not but are useful, which are not useful, this would then be used to guide development of the collaborative features of the new tool.

The analysis of these software can be found below. An important area that all the reviewed software were lacking in is collaboration. In recent years cloud software has taken off and made real time collaboration a possibility. The current work flow using the existing software is do your analysis, save it, email it to a colleague with additional files and notes, have them open it and try and work out what is happening. It is a disjointed conversation. None of the tools surveyed offered a more efficient approach.

- GoogleDocs - Todo
- Apache Wave - Todo
- Pidoco - Todo
- Lucid Chart - Todo
- SubEthaEdit - Todo

This tool has a space in the existing landscape in two ways. The first as offering visualisation capabilities similar to those in the other biological modelling software to Bio-PEPA. The second is by implementing modern collaboration features, making it a unique tool.

1.2 Previous Work

Early on in the first stage of the project it was decided to separate this project from the Eclipse plugin. It was felt that Eclipse is not the right tool to do data visualisation in.

The initial development stages were focused on getting the new tool from zero functionality to matching the visualisation features of the Eclipse plug-in. This involved writing an early version of the U.I., parsing the Bio-PEPA results data, and plotting it using matplotlib.

After matching the existing functionality it was time to add new functionality. The first new feature was intensity plotting where the colour of the line increases in intensity/opaqueness as the population of the species increases. Then visualisation of the model was added. It used a system of nesting circles and rings to build a heirarchical view of the cell from its model components. Finally the user was given control of the plot, allowing the toggle whether it is shown or not, how it is plotted, what colour it is and the thickness of the line.

Over the course of the first stage of work a number of evaluations were carried out with potential and actual users of Bio-PEPA. The findings from these evaluations were used to improve the tool.

1.3 Results

Break down by project stage? Aniticipating this to be 1.5 - 2 pages

2 Goals

Whilst researching the problem and the existing software, and during the first stage of development a number of goals were identified.

- Improve existing capabilities
- Visualise the model more intuitively
- Visualise closer to the cell level
- Animation of the data
- Investigating which combinations work best
- Add the ability to annotate the visualisations
- Allow the ability to save and load the program state
- Provide a full session history to the user
- Data Mining
- Making meta data accessible

During the first half of the second phase of development new goals were identified.

- Google docs style collaboration.
- Searchable database of time series plots.
- Manipulation and Exportation of data.

2.1 Previous Work

Could move here from intro, could talk a bit more about how they could be improved.

2.2 New Work

For each of the below, why, how inc conceptual problems, impact, how it could be improved

2.2.1 Annotation

Annotation of the Graph Annotation of the graph was an important feature to add. It is one of Grinstein's features that data visualisation software should have. This is because these data visualisation should help analyse. If you had a print out of a graph it would be second nature to draw over it, highlighting areas of interest. It needs to be able to be performed digitally as well. As well as helping users analyse their data it also means that images for presentations can be prepared without having to save the graph and open it in an external program. If you did this and then wanted to change the graph you would have to re-annotate it. This is wasted time and frustrating. Being able to do it from within the visualisation problem. Another benefit of being able to annotate is it is another way of attaching additional information to the graph when sending it to a colleague, having this information on the graph saves them from flicking back and forth.

Annotation on the graph was relatively straightforward. matplotlib provides annotation capabilities in two forms. One form is annotating arrows with or without text, and the other is arbitrary drawing on the graph.

Buttons for each of the annotations have been placed on the matplotlib toolbar. There were problems in making it usable. Text and Circles were usable as all they require is one click. However the arrows required two clicks. At first it was similar in that it required two clicks, first is the start point, second is the finish point. This was non obvious. It was changed so that the first click still placed the start point, after the first click, now however, a temporary annotation is continuously redrawn that has its end point of wherever the mouse is. This allows the user to see the arrow they are drawing.

Next the annotations had to be able to be edited or deleted. They could not just be clicked as they are not a U.I. element. The solution to this was to have an array of annotations. When a user right clicks on the graph it searches through all annotations and selects the annotation that was closest to the click, if it was below a certain threshold. The selected annotation is then highlighted red, and context menu appears to give feedback to the user that they have successfully selected. Then properties of the selected line can be changed.

Annotation of the Animation After completing annotation of the graph it was important to expand this to other areas of the program, particularly the animation panel. This poses more of a challenge than for annotation of the graph. There are a number of issues to overcome.

1. How exactly to do the annotations? For the graph matplotlib has built in annotation support. wxPython does have drawing support but no in built annotation support. This means manually handling the drawing on top of the animation visualisation. Manual

drawing means that the automatic layout functionality that wxPython provides has to be left behind.

2. When to display annotations? When an annotation is drawn on the graph it is there to be displayed at all times. The appearance of the graph does not change over time. However on the animation visualisation appearance does change over time. The problem faced when annotating is then is whether to have annotations available at only specific times in the animation, or to have them there the whole time, and if they are going to appear and disappear how can it be done without being distracting.
3. Control over annotations – Related to the previous item, when a user wants to edit or delete an annotation on the graph it is always there. However on the animation panel if the annotation is temporal, then it is not always there for the user to edit or delete, and it would be frustrating for a user to constantly have to search through the animation time to look for annotations to change them.

WRITE MORE AFTER DOING THIS.

2.2.2 Animation

Animation was a key goal of the project. The core aim of the project is to help biologists who aren't comfortable with traditional time-series graphs. So the aim was to give them visualisations closer to what they see in their domain. This has been accomplished by displaying spatial data in the shape of the cell.

The animation is ideally used to display species moving through a cell. This collapses what would be multiple different lines, that give no indication of their real position in the cell, into a single image. There is one segment in the cell animation for each line. The colours in each segment reflect the colour of the line on the graph. Then over the course of the animation the colours are set by the colour in the intensity plot version of the line. This allows the researcher to compare the two visualisations and will hopefully help build their confidence on the graph plot.

2.2.3 Data Mining

2.2.4 Search

2.2.5 Collaboration

2.2.6 Usability

In all the evaluations of the project so far users have commented on the difficulty of using parts of it. Action has been taken to make it easier to use. Many of the changes have been guided by Shneiderman, Norman & Nielson's guidelines. Specifics are detailed below.

Undo & Redo Shneiderman calls for easy reversal of actions and Nielson calls for user control and freedom – an emergency exit from an unwanted state. For this undo/redo functionality has been added. This required refactoring of the project code, so that the raw data is in one location, inside a singleton, any changes to this data model and the next U.I. update uses this

data to display the visualisations. This data model is stored as a dictionary. To implement undo and redo copies of the data dictionary are pushed and popped onto the stack. Copies are pushed onto the undo stack on any atomic change the user makes. This gives the user a full session history to go back through and this was one of the early goals from the first project phase.

A problem was encountered when trying to copy the dictionary onto the stack. When just pushing the dictionary onto the stack it would not put a new copy onto the stack, so any changes to the dictionary after it has pushed onto the stack are also there in the stack. Python dictionaries have a copy method. Copy only does a shallow copy – any objects in the original dictionary will have their reference placed in the new dictionary. This was fine for some parts of the session dictionary, but others it was not, in particular lines and annotations. This was solved by using deep copy. With deep copy a new copy is made of objects as well. Some elements of wxPython and matplotlib were unable to be deep copied, but this was fixed whilst focusing more around around the data – so the U.I. elements use the data, not the other way around.

Saving It is important that a user is able to save and load the visualisation session. Because the user is able to add annotations and change preferences and attach files they do no want to have to repeat all these actions every time. Luckily it was able to build on the reworking done to get undo/redo to work. Although further work was required. Python has a module called pickle to serialise and deserialise data. When saving the data dictionary is pickled and written to a file and the loaded the reverse happens. Because the program is now focused on the data model once a previous session has been loaded, a U.I. refresh is triggered and the loaded session is visible.

Saving the data also enables limited collaboration. The user can annotate, change colours etc, save the state to a file, and then email that file to a colleague. They can then load this file, and see what the user was seeing. They can then add more annotations, correct or delete existing ones. This can then be saved and emailed back. This is useful – better than no collaboration. But it happens asynchronously.

Feedback Norman and Shneiderman both call for feedback to be given to the user so that they can be sure that an action has been accomplished. This feedback can come in a number of different forms, and was in place in some parts of the project. Existing feedback in the project was a natural byproduct of some of the features. For example when loading a results file the feedback that the load has been successful is that a graph appears on the screen, if it doesn't succeed then something has gone wrong. Additional feedback has been added to the project, when adding annotations the cursor changes to indicate to the user that they can interact with the graph in a different way. And before there was no indication that a save operation had been successful. Now the title bar text changes to display "unsaved" when the user makes a change and then changes back to "saved" when a successful save has been performed.

Guiding the User The first evaluation of the second phase of the project unearthed that the users struggled when there were multiple ways of doing the same operation, but that had slightly different uses. These multiple paths have been removed. For example now there is only one way to open results files initially. To help guide the user further U.I. elements are enabled and disabled as appropriate. So when the program is first loaded the only action a user can

perform is to load a session or start a new session. Afterwords other U.I. elements are enabled to allow them to start using the tool effectively.

2.2.7 Data Manipulation and Export

3 Work Done

This section describes the work done in the second phase of the project

4 Evaluation

4.1 First Evaluation

The first evaluation in the second phase of the project occurred just before the halfway mark. The group was made up of two people. One who had taken part in the first evaluation meeting and one person who had no knowledge of the project.

4.1.1 User Group

As I did not have a domain expert available I was not able to do insight based evaluation. I took a more traditional approach. Before the evaluation I prepared a typical scenario that a user might encounter. The task was to open a file, annotate it and play the animation, and attach some supporting documentation. The task was prepared at two levels of breakdown. One was a paragraph of text at quite a high level. The second level of breakdown was a step by step instruction of each action to perform. I then observed them as they attempted the task and offered assistance when required. Afterwards I gave them a questionnaire to fill in about their experience, afterwords we went through and discussed their answers and any further thoughts that they had.

The task was prepared at two levels to try and gauge how easy the program is too use. The users were presented the textual description and if they had struggled they would have been given the step by step instructions instead. The users were able to complete the task from the textual description alone. This is a good sign that the new tool is usable.

Some issues were encountered:

- Being unfamiliar with MacOS – Both users were unable to locate the menu bar as it is not attached to the program as in Windows. Future evaluations will use Windows.
- When annotating they were unclear as to what was going to happen when annotating. For example when annotating the graph with an arrow the user was just left to click twice, with no indication of what would happen. This has now been fixed, different cursors are used to give feedback to the user that they should click, and rather than just relying on two clicks with no information as to where the arrow is going to point, after the first click (which places the tail of the arrow) an arrow will be drawn that follows the cursor until the second click placing the annotation.

- Lack of ability to edit, move, or delete annotations – Once an annotation was placed it was there for good. The ability to edit annotations was always planned, it just had not been implemented in time. But the amount of frustration it gave the users was very high. It was a principle in all three of the design lists that a user should be able to fix mistakes. Since the evaluation editing and deleting of annotations have been implemented. This means any mistakes can be corrected.
- Initially they were confused by what all the buttons on the matplotlib toolbar did. After discovering the tooltips and seeing what effect the buttons had they were comfortable with them. Also all the matplotlib built in buttons on the toolbar are undoable from the toolbar. Annotations are editable and deletable and also covered by the undo functionality implemented across the whole program. Being able to recover from their actions on the toolbar means no hindrance to discovery and so needs no further action.
- The users were confused by some of the terminology. In particular "save graph" and "save model". These items in the menu have grouped more carefully to help the user distinguish them. A related issue was worrying that "save graph" was going to override the results file. To rectify this the menu items that create new files have been renamed "export ...".
- When creating a session they stuggled, they did not know what the title was referring to. And when trying to add files, rather than use the add files button in the dialog, they tried to use the file menu. Having two routes into doing the visualisation seemed to be confusing them. Now the file menu open file has been removed. To create a visualisation the user has to go through the new session wizard.
- When placing species in the cell one of the users did not understand what they were being asked to do. One of the users did understand. To fix this user input has been removed from the equation. This has required the model file to also be chosen, but then species locations are parsed automatically.
- They liked the animation feature and thought it would be very useful (One of the users did their phd in transport and expressed a desire to have had this feature during the phd). They did feel that it wouldn't be useful directly for papers, but that it could be used to inform the content of papers.
- One of the users asked if they had a map of the cell. When presented with the model visualisation they thought that it did look nice, but were unsure of its usefulness – HAS IT BEEN TAKEN OUT.
- The results from the survey indicated that both users thought the tool had a good appearance. Was average in difficulty to use – neither easy nor difficult, the annotation buttons on the toolbar were clear as to what they did, that it was obvious how to attach supporting files. Both users thought that it is bery useful to attach files to the session so that they can be easily emailed to a colleague. When asked about having the graph be automatically annotated they though it would be very useful but wanted to the ability to disable them. Both found the animation useful.

4.1.2 Personal

At this stage in the development the program was in a state where some existing functionality had been broken and gone unnoticed during the implementation of the new features. This showed some bad architecting that had found its way into the code. There were multiple paths

through the program that data was taking, and duplicated code in places. Since then a majority of these bugs have been ironed out, and the code much better architected and the duplications removed. At the time of the evaluation with the users not all the features could be tested with them – mainly the plot preferences dialog. These features have now been fixed and they will be evaluated by the users at the next meeting.

Having the users use it has also highlighted a number of usability problems, menus being badly organised and named, features, such as annotation, relying on knowledge of how they work to use them. All this created a unfriendly environment for the user. This was due to losing sight of the need for usability during development and when testing new features not removing the knowledge of the code from my mind. Since this evaluation the three lists of usability have been read and the code gone through and had the principles applied. TALK A BIT MORE HERE ABOUT THIS.

I am pleased with the positive feedback on animation and annotation – two of the core new features.

4.2 Evaluation 2 - Start of Second Semester

4.2.1 User Group

4.2.2 Personal

4.3 Evaluation 3 - End of Second Semester

4.3.1 User Group

4.3.2 Expert User

4.3.3 Personal

5 Conclusion

5.1 Comparison to Objective

5.2 Challenges Faced

5.3 The Future

5.4 Final Remarks