A Tool for Visualising Cell Model Results
MInf Project Phase 2

James Hulme
s0901522

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2014

Abstract

his is where the abstract will go

Table of Contents

Chapter 1

Introduction

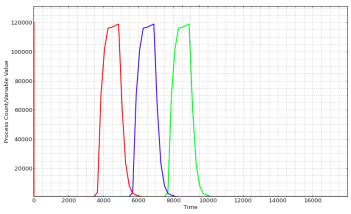
1.1 Introduction

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 the tools they use to help them relate the data to their field by incorporating domain knowledge.

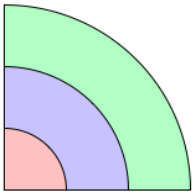
One such tool that can be used for modelling is Bio-PEPA, an extension of the PEPA process algebra. Bio-PEPA is currently implemented as a plugin for the Eclipse IDE. Bio-PEPA visualises the model results as time-series graphs. There is one team of Src researchers in particular, who use Bio-PEPA and do not have the numerical confidence, as described above, to be comfortable using Bio-PEPA. This team was the original focus for the 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 1.1). 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 are too abstract. It requires the use of a biological metaphor for spatial information to be easily interpreted. In this case using a visualisation based on a cell can more intuitively show how the species moves through the cell. It is this type of inference that the Src researchers find challenging to do with Bio-PEPA currently.

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 objective changed (to



(a) Time Series Representation



(b) Spatial Representation

Figure 1.1: One species at three locations in the cell represented traditionally on a time series graph and also spatially in a cell

reflect that the project was about delivering a finished program and not about the results of many experiments) 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.

1.1.1 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 packages reviewed at the start of the project 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 could 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 had 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 types 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.

As the project scope expanded to include goals not specifically related to visualisation it was prudent to perform another software review covering the new features, in particular software that allowed for collaborative editing. It is important to see what features are commonplace, which features are not commonplace but are useful, and which features are not useful. This analysis would then be used to guide development of the collaborative features of the new tool.

Three products were chosen for review: Google Drive, Pidoco and Lucid Chart. Analysis of these software can be found below.

1.1.1.0.1 Google Drive , which was previously Google Docs, is one of the most widely used collaborative pieces of software by a variety of user types. The focus of the review was on the word processor. Of the collaborative software reviewed this was felt to be the most user friendly. One of the nicest features was a cursor indicating where every user currently editing the document is typing. Each user is also given a different colour allowing you to know who is doing what in real time. Many people can edit a single document at once. As well as editing documents users can also leave comments on the document. Changes made by users appear near instantly to all users, the speed of editing is very important as it is frustrating as a user to have to wait to see changes others are making. It is also very easy to invite others to edit the document with you. Each document has a unique link and if a user visits that link they are taken to the document and can start editing. Different permissions can be granted to different users allowing some level of collaboration with people who you don't necessarily want to grant full write access, these users can then just look at it in real time and offer comments. Different parts of the editor have different levels of collaboration. All text that is changed is changed for all, but preferences like font choice are only changed for the user, unless another user edits any text. Also importantly from a User Experience (UX) perspective is that any conflicts that arise appear to be resolved without any user intervention. A history of what each user has done to the document is also provided and any unwanted changes can be rolled back.

1.1.1.0.2 Pidoco is a collaborative wireframing tool. It was not as user friendly for collaboration as Google Drive. Pidoco is much less instant. Sometimes manual refreshes were required to display the work the other users had done. Pidoco also supported multi user editing, however there was no way of seeing which users were editing a document. There was also a history of what changes each user had made. Pidoco has no messaging or commenting system which makes asynchronous collaboration more difficult. Sharing the work requires an email to be the sent to the user, they cannot simply be given a link. Again different parts of the workspace have different levels of collaboration. Any User Interface (UI) widgets that are placed are shared, but if one user zooms in on a particular area that other users are not forced to that zoom level.

1.1.1.0.3 Lucid Chart is a collaborative diagramming tool. Lucid Chart lies between Google Drive and Pidoco in terms of collaboration speed. It is not as instantaneous as Google Drive, but it does not require periodic manual refreshes like Pidoco.

It also allows multi user collaboration and documents can be shared by link or email. Users can be granted read or read and write permissions on the document, like Google Drive, so you can collaborate with people you don't want to be able to fully edit. It has a chat system and users can comment on the document making asynchronous collaboration easier. Lucid Chart does not let you see what a user is doing in real time, but it does provide a full revision history so you can see what changes each user has made. Lucid Chart is different in that it appears to be fully collaborative. Even font preferences are synced across users, if one user clicks bold all users will start typing in bold.

1.1.1.0.4 Findings: By implementing more advanced visualisation features this tool makes Bio-PEPA a more attractive modelling tool, by bringing the feature set in line with the alternative modelling tools. There are also features that are not offered by any of the other reviewed biological software – visualisation and non visualisation features. None of the other tools gave the users as much control over the look of their results. Also none of the other tools had the correspondence between the different visualisations that exist in this new tool. This tool is also the only one that allowed the user to annotate and attach supporting documentation. None of the other modelling or visualisation tools reviewed offered any sort of collaborative features - real time or not. This makes this tool unique and innovative.

1.1.2 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 having zero functionality to matching the visualisation features of the Eclipse plug-in. This involved writing an early version of the UI, parsing the Bio-PEPA results data, and plotting it using matplotlib.

The next stage of the project was to extend the 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. The next feature added was visualisation of the model. Model visualisation used a system of nesting circles and rings to build a hierarchical view of the cell from its model components. Finally the user was given control of the plot, allowing them to alter whether lines are plotted or not, what colour the line is and the thickness of the line. Figure 1.2 shows how the main screen of the program looked at the end of the first stage of development.

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.

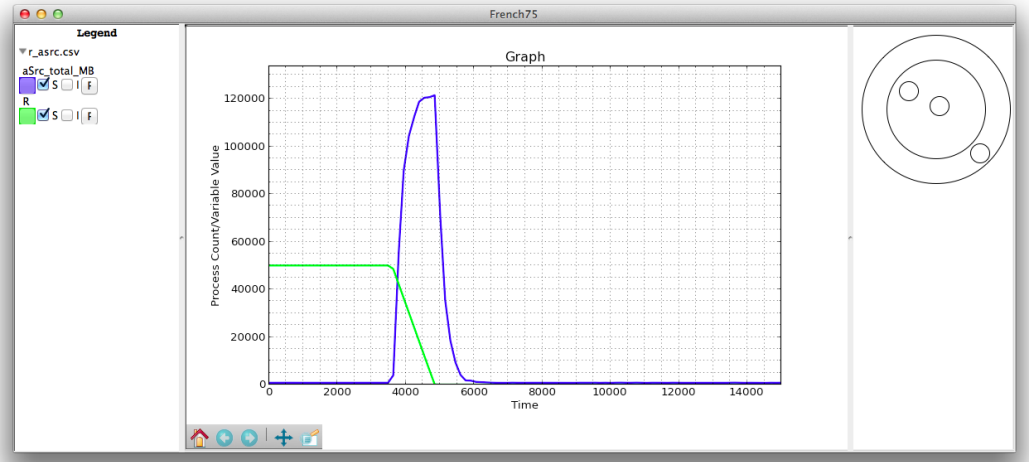


Figure 1.2: Main Screen of the Tool

1.1.3 Results

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

Chapter 2

Goals

Signpost text

2.0.4 Previous Goals

Through research into the problem a number of goals were identified. After performing a review of the existing software the goals were refined. The goals were again refined during the first stage of development. Some of these goals were completed in the first phase of development:

- Improve existing capabilities – This goal was completed in the first phase of development, with the implementation of the intensity plotting and the extra customisation available to the user.
- Visualise the model more intuitively – This goal was completed in the first phase of development with the implementation of the hierarchical drawing of the model components.

Others were to be completed during the second phase of development:

- Provide visualisations that take into account the domain - This has been implemented in the second phase of the project. This goal was merged with animation of the data.
- Animation of the data - This has been implemented: the user can now visualise species moving through a cell.
- Investigating which visualisation combinations work best - This was planned to be performed in this stage of the project, but due to the new features not all being different visualisations the goal is less relevant now, and has been removed.
- Add the ability to annotate the visualisations - This goal has been implemented: the user can add annotations to the visualisations.
- Allow the ability to save and load the program state - This goal has been implemented: users can save and load the program state into files, these can then be

emailed to other users who can modify them.

- Provide a full session history to the user - This has been implemented, the user has a full undo and redo history within the session allowing them to easily correct mistakes.
- Data Mining - The plan for this has not changed since the first phase.
- Making meta data accessible - The plan for this has not changed since the first phase.

2.0.5 New Goals

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

2.0.5.1 Data Manipulation and Export

This goal was added after a meeting with the maintainer of BioDARE. This web based tool is aimed at results from laboratory experiments, specifically experiments relating to circadian rhythm.

The maintainer explained that he had found that researchers often visualise their data after it has been normalised. This removes any issues where different species have different scales and makes it easier for the users to interpret the data.

BioDARE can also export the raw data allowing the researchers to use it in other applications if they desire.

This seemed like a very useful feature for users and it was added as a project goal.

2.0.5.2 Time Series Data as a Query

There has been a lot of research in recent years about using time series data as a query against a database of other time series plots through some similarity measure.

This would be a very useful feature to add to the tool. It would allow researchers to search through their past work for similar time series data that also shared some species. They then know that different species behave similarly when exposed to another species.

Eventually this database of plots could be a central repository online available to all researchers, allowing users to see if their research overlaps with any other research – identifying potential collaborators.

2.0.5.3 Real Time Collaboration

An important area that all the reviewed software were lacking in is collaboration, real time or not. 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. A new goal for this project was to offer real time collaboration to improve the existing work flow.

Chapter 3

Work Done

3.0.6 Previous Work

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

3.0.7 New Work

Signposting text

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

3.0.7.1 Annotation

3.0.7.1.1 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 data visualisations should help users analyse their results. If you had a print out of a graph it would be second nature to draw over it and highlight areas of interest. This task needs to be able to be supported digitally as well. As well as helping users analyse their data, being able to annotate 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 frustrating for a user and wastes their time. Being able to do it from within the visualisation solves this problem as the raw data and the annotation data are together. Another benefit of being able to annotate is having 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 from an email or other supporting documentation.

Annotation on the graph was relatively straightforward to implement. 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.

Users of the new tool are provided with four annotation types: arrow, text, arrow with text and circles. Buttons for each of these annotation types have been placed on the matplotlib toolbar.

There were problems in making annotation user friendly. Text and circle annotations were intuitive as all they require is one click – click where you want the annotation and it will be placed there. However the two arrow annotation types required two clicks. The first click is the start point (tail of the arrow) and the second click is the finish point (head of the arrow). This was not obvious, when handed over to the users they didn't know that it required two clicks and didn't know whether the arrows would be drawn head to tail or tail to head. The technique for placing arrows was changed so that the first click still fixed the position of the tail of the arrow. However the behaviour after the first click has changed, now a temporary annotation is continuously redrawn that has the head of the arrow 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. The annotations can not just be clicked as they are not a UI widget like a button. 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 a context menu appears to give feedback to the user that they have successfully selected an annotation. The context menu then gives the user the option to edit or delete an annotation. Editing an annotation only allows for editing text. For changing position the annotation has to be deleted and redrawn.

3.0.7.1.2 Annotation of the Animation

After completing annotation of the graph it was important to expand this to the animation panel, as this is the other area where visualisations are put. This posed more of a challenge than for annotation of the graph and there were a number of issues to overcome.

1. How to implement the annotations? For the graph matplotlib has built in annotation support. wxPython does have drawing support but not in built annotation support. Annotating on the animation will need manual handling of the drawing on top of the animation visualisation. Manual drawing means that the automatic layout functionality that wxPython provides cannot be used.

2. When to display the annotations? When an annotation is drawn on the graph it is displayed at all times. The appearance of the graph does not change over time. However the appearance of the animation visualisation does change over time. The problem faced when annotating 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. How to give the user control over the annotations? 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 visible for the user to

edit or delete and it would be frustrating for a user to constantly have to search through the animation to look for annotations to change them.

3.0.7.2 Animation

THIS NEEDS A LOT OF WORK

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 goal was to provide them with 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 in a graph, 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.

To make setting up the animation user friendly to control required the model file so that the location hierarchy could be parsed. Before this there was an awkward system where the user had to input where in the cell a species is. This was time consuming, awkward and quite brittle. At the time it assumed that there would be three compartments in the species, which is a terrible assumption to make.

The requirement of the model file for parsing animation has also led to animation replacing the previous model visualisation. In the set up phase after the the model and the species have been parsed the user is presented with a cell segment, similar to what is seen in the animation panel. The cell segment is split into different regions, one for each region of the cell. These segments are then coloured if the selected species is present in them. The user can select between all species in the selected results files. This has a number of benefits. First, they can sanity check that they have matching results and model files. Second, they can see how the model is structured.

Similar control is provided on the animation panel itself. The user can see the animation focused on a specific species, in which case a cell segment is drawn for each file that the species is in. Or they can have the animation focused on a specific results file in which case a cell segment is drawn for each species in that results file.

3.0.7.3 Data Mining

3.0.7.4 Search

Using a time series as a plot posed some interesting problems.

- How to cope with different scales

- How to cope with events happening at different times
- How to represent the plot to allow for efficient search
- How to determine similarity between two graphs

All of these needed to be overcome for this feature to be useful.

This is an area of active research. Early techniques used simple techniques such as Euclidean distance, but these simple approaches gave poor results. It is feasible to have a species that exhibits a similar reaction to our query (the line has the same shape) but it happens at a different time in the experiment than our query plot. In other words the graph is offset. We can tell that the two graphs are similar, but euclidean distance will return that they are unsimilar. The same problem occurs with differences in species population giving an offset in the x axis.

After researching current techniques an approach was taken that solves all the problems above.

The first step is to convert the input data to be a list of all n length sub lists of the input data. i.e. with input data [1,2,3,4,5] and n = 3 our input data would become [[1,2,3], [2,3,4], [3,4,5]]. The sub lists are our features. We then normalise each feature to be zero mean, unit variance.

The next step is to convert the continuous data into discrete data and reduce the dimensionality of it. By normalising the data to be zero mean and unit variance we have allowed a normal distribution to be easily fitted to our data. Using the normal distribution we convert a data point in our sub list to be one of three possible values. Here we use the characters ‘a’, ‘b’ & ‘c’. This builds a string representation of the plot.

The paper this approach is taken from then calls for reducing local duplicates. This further reduces the size of our fingerprint and increases the efficiency. Removing the local duplicates means any ignoring runs of a certain string can be replaced with just one instance of the string.

This provides us with a finger print of the plot. A string size of 8, with 3 possible value leads to a vocabulary size of 6561 possible representations. A single plot will have very few of these, so we use a sparse representation. For each line a dictionary is stored. The dictionary keys are the strings in the fingerprints and the value is the count of that string in the plot’s fingerprint.

This format of the fingerprint also solves the problem of similar features happening at different times. The plot fingerprint is a bag-of-words, in this case a bag-of-features. There is no order to them, it is as though we are assuming that all features appear at the same instant.

Now that we have a representation of the plot that is scale and time invariant and allows for efficient comparisons we can find a method to calculate the similarity.

We could simply just use Euclidean distance again, but this does not do anything to weight rarer features that can tell us more about a graph. An alternative distance measure would be cosine distance, but this suffers the same drawback.

It was decided to implement tf.idf weighted cosine as the similarity measure. This takes into account term frequencies to provide a higher weight to rarer vectors. tf.idf is very well researched in the field of information retrieval and text mining. Given the representation of the finger print is equivalent to a bag-of-words it seems sensible to apply tf.idf to this domain.

3.0.7.4.1 tf.idf is a way of ranking how important a word is. Taken into account are frequency of a word in the query, frequency of a word in the document, length of the document, average length of documents in the corpus, total number of documents and frequency of the word across all documents.

The formula is: INSERT FORMULA.

These term weights are then used in a tf.idf weighted cosine.

This approach allows for effiecient search. If we have two indexes. One where keys are the documents and the values are the words in the document and their frequencies. Then we have another index, an inverted one. Where the keys are the words and the values are a list of documents that contain them. The individual components of the tf.idf equation can therefore be calculated with very little effort. This means that a plot can be compared to the database of plots in O(n) time.

Initial results are very promising with the tf.idf weighted cosine providing much clearer results than with simple euclidean distance. There is not a large enough truth set of plot similarities to be able to confidently say that it is effective. However these early promising results seem to indicate that further research would certainly be worthwhile, but outside the scope of the project.

DISCUSSION OF THE EARLY RESULTS

3.0.7.5 Collaboration

The initial step of allowing collaboration is enabling two instances of the program to talk to each other. There are a number of techniques to allow for this. Data could have been sent directly through sockets, another option would be to create a web server and send get requests and pass parameters. It was decided to use an existing library to handle the communication. The library chose was simplexmlrpc. This allows each instance to run a client and a server. The server has an api and this can be called.

On initial connection the ‘master’ sends its entire session state the the ‘slave’ uses this as its initial state. After this whenever either client performs an action it issues a call to the other clients server to perform the same action.

An issue that was encountered is how to ensure that both clients are seeing everything in the same order. Lamport clock proved to be the solution.

3.0.7.6 Usability

In all the evaluations of the project users have commented on the difficulty of using parts of the tool. 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.

3.0.7.6.1 Undo & Redo Shneiderman calls for easy reversal of actions and Nielson calls for user control and freedom – an emergency exit from an unwanted state. To address this, an undo/redo functionality has been added. This required refactoring of the project code, so that the session data is in one location, inside a singleton. Any changes to this data are picked up during the next UI update and are reflected in the visualisations. The session data 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 of it 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 for others it was not. In particular, lines and annotations, which are custom objects presented problems. 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 the data – so the UI elements use the data, not the other way around.

3.0.7.6.2 Saving It is important that a user is able to save and load the visualisation session as they may not be able to complete all their analysis in one sitting and may want to come back to their work in the future. Without the ability to save and load the user would have to repeatedly add annotations and change preferences and attach files. It was possible to add Saving and loadingby building on the work done to implement undo & redo, although further work was required. Python has a module called pickle to serialise and deserialise data. When saving, the dictionary containing the session data is pickled and written to a file and when loading the reverse happens. Because the program is now focused on the data model, once a previous session has been loaded, a UI refresh is triggered and the visualisation reflects the loaded data.

Saving the data also enables limited collaboration. The user can customize the appearance and add annotations. They can then save the state to a file and email that file to a colleague. The colleague can then load the file and see the user's work. The colleague can then correct any issues and add their own work. The colleague can then save this

and email it back to the user. This is useful and is better than no collaboration, but it is entirely non realtime.

3.0.7.6.3 Feedback Norman and Shneiderman both call for feedback to be given to the user so that the user 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 already.

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 operation has been successful is that a graph appears on the screen. If the graph does not appear 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.
- 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.

3.0.7.6.4 Guiding the User The first evaluation of the second phase of the project unearthed that the users struggled to choose the correct action as there were multiple ways of doing the same action that had slightly different use cases. There was also confusing language in the menu options. 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 UI elements are enabled and disabled as appropriate. Now when the program is first loaded the only action a user can perform is to load a session or start a new session. Afterwards other UI elements are enabled to allow the user to start using the tool effectively.

3.0.7.7 Data Manipulation and Export

3.0.8 Finished Product

Overview and walkthrough of tool

Chapter 4

Evaluation

4.1 Evaluation

4.1.1 First Evaluation

The first evaluation in the second phase of the project occurred in November 2014. The user 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.

4.1.1.1 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 run the animation visualisation, and attach supporting documentation. The task was prepared at two levels of instruction. The first level was a paragraph of text that described what was to be done. The second level was a step-by-step list of instructions to perform. I observed the user group as they attempted the task and offered assistance when required. Afterwards the user group was given a questionnaire to fill in about their experience. After filling in the questionnaire 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 to use. The users were first presented with the textual description and if they had been unable to complete the task with this then 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:

- The users were 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.

- The users were unclear as to what was going to happen when annotating. When annotating the graph with an arrow the user has to click twice to place it but there is no indication of this, nor was it clear to them which way the arrow would be drawn. 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 permanently. The ability to edit annotations was always planned, but 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 Norman, Neilson and Schniederman's 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. If a user were to do something they did not intend they are able to undo it. All the matplotlib built-in buttons on the toolbar can be undone and redone from the toolbar. Any buttons implemented for this tool are covered by the undo and redo 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. It would be desirable to have the two undo methods unified but a way to do this could not be found.

- The users were confused by some of the terminology. In particular "save graph" and "save model". These items in the menu have now been 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 ...".

- The users struggled to start a new session. When asked for a title they did not know what the title was going to be used for. When trying to add files, rather than use the add files button in the dialogue, they tried to use the file menu. Having two routes into 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 would be useful when deciding what to include in a paper.

- One of the users asked if there was a map of the cell. When presented with the model visualisation they thought that it did look nice, but they were unsure of its usefulness. The model viewing has since been merged into the animation visualisation.

- The results from the questionnaire indicated that both users thought the tool's appearance was good. The tool was average in difficulty to use – neither easy nor difficult. The annotation buttons on the toolbar were clear as to what they did. It was obvious how to attach supporting files. Both users thought that it is very useful to attach files to the session so that they can be easily emailed to a colleague. They thought it would be useful to have the graph automatically annotated, but they wanted to the ability to disable any automatic annotations. Both users found the animation useful.

4.1.1.2 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 highlighted architectural flaws in 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 duplications removed. The code much better architected. At the time of the evaluation with the users not all the features could be tested with them – mainly the plot preferences dialogue. These features have now been fixed and they will be evaluated by the users at the next meeting.

Having the users use the program has also highlighted a number of usability problems: menus being badly organised and named, features such as annotation rely on assumed knowledge to work them. All this created an 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 refocused on and the code gone through and the principles applied.

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

4.1.2 Evaluation 2 - Start of Second Semester

4.1.2.1 User Group

4.1.2.2 Personal

4.1.3 Evaluation 3 - End of Second Semester

4.1.3.1 User Group

4.1.3.2 Expert User

4.1.3.3 Personal

4.1.4 Overall Self Evaluation

Scribble over lots of stuff talk about changes

Chapter 5

Conclusion

5.1 Conclusion

5.1.1 Comparison to Objective

5.1.2 Challenges Faced

5.1.3 The Future

5.1.4 Final Remarks