University of Glasgow | School of Computing Science

A Hands-on Approach to Learning Molecular Biology Techniques

Ross Eric Barnie
Dmitrijs Jonins
Daniel McElroy
Murray Ross
Ross Taylor

Level 3 Project —

Abstract

Abstract-like things

Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name:		Signature:	
Name:		Signature:	
Name:		Signature:	
Name:		Signature:	
Name:		Signature:	
Name:		Signature:	

Contents

Chapter 1

Introduction

1.1 Preliminaries

There are some terms that will be used later in the report that should be clarified now, so as to avoid confusion. The aim of the project is to creating a teaching tool for PCR, or Polymerase Chain Reactions. This is the process of amplifying a sequence of DNA thousands to millions of times. It should be explained that DNA sequences are made up of two strands, comprised of bases of the nucleotides Adenine, Thymine, Guanine and Cytosine, represented by the letters `a`, `t`, `g`, and `c` respectively. Base pairing is when one base bonds with its complement on the other strand. `a` and `t` complement each other, and G and C complement each other.
Primers, used to select the sequence for PCR in a given selection of DNA, are shorter fragments of DNA, usually between 20 and 30 bases in length. For use in PCR, a primer must be chosen from the left of one strand (this is the forward primer) and the right of the other (this is the reverse primer), and these must obey a number of rules, which are the focus of the teaching tool:

- The primer must not self-anneal. This means that if the primer were to fold in on itself in any way that more than 3 bases in a row on one side paired to the base on the opposite side, the primer would fold over and become useless.
- The melting temperature, calculated in degrees Celsius using a simple mathematical formula involving the frequency of `a`s, `t`s, `g`s and `c`s, must be between 50 and 65C, and within 2-3C of each other.
- It must be unique within the strand.
- The percentage of Gs and Cs within the sequence must be between 40% and 60%.
- The length should be between 20 and 30 bases.
- The same base should not be repeated several times in a row.
- The last base of the primer should be either a `g` or a `c`.
- The primers should not anneal to each other. This means that the rule is broken if, at any point in the overlap of the primers, more than 3 bases in a row paired with the overlapping primers base.

It should also be explained that we are developing the teaching tool in Java, using Netbeans, a free IDE primarily designed to be used with Java, and developing the user interface with Swing, the primary Java GUI widget toolkit.

1.2 Aims

The overall aim of this project is to produce a piece of software to help Molecular Biology students learn about PCR and Primer Design Techniques and to allow them to test their knowledge of these subjects. Initially, this overall aim was divided into key tasks to be completed and important aspects of the interface design to be implemented:

1. The software should work as an interactive tutorial which users can work through. This requires:
   - A number of areas for users to enter their own choice of data e.g. choice of primers. For this feature to be useful as an educational tool feedback must be provided upon data entry.
   - Users should be able to experiment with different data e.g. examining the different melting temperatures of different primers. This requires the ability to easily move forwards and backwards between the different stages of the tutorial.
   - To help newer users and students who are unfamiliar with PCR there should be simple instructions to tell users what to do on each page. There should also be a page displaying the rules of PCR and primer design which should be available at all times.
2. The software should be useable by all users, regardless of their different levels of knowledge, ability etc. This includes:
   - To achieve this, the interface should be uncomplicated and intuitive without compromising the required functionality. This will be aided by the instructions and help section mentioned above as well as labels placed next to any areas users can interact with.
   - Any section which makes use of colour should be designed with colour blind users in mind.
3. The software should improve upon the tools currently available for learning primer design. The main issues with these systems are:
   - The low level of interactivity offered by the systems. Users who are not actively working through a tutorial or a demonstration are likely to lose interest faster so it is important to make them involved with every step of the tutorial by having them design their own primers etc.
   - The available tools rarely go into detail about primer design specifically. Therefore, an important aim for the project is that primer design must be explained in high detail and provide enough information to be informative, whilst remaining interesting to students using the system.

4. Another aim related to accessibility is that the users should be able to download and use the software from home. This means that the program must be able to run on a variety of different operating systems and computers with varying performance levels. With this in mind it was decided that the program should be written in Java due to it being highly portable.

1.3 Background

1.4 Motivation

In order to understand the motivation for the development of the system, we were sent three links to systems currently in place which attempt to make learning this process more interactive and/or visual. However, videos and multimedia in general have been questioned as teaching aids in the past. Simply because the information is in video or multimedia format does not necessarily mean that it is benefitting the learning of its viewers, or creating the correct environment to encourage learning. Interactivity, along with other factors, are key to engaging people to learn (DeKanter, 2004).

The first was a video hosted on YouTube ('dwildridge', 2012), made by demonstrators within the School of Life Sciences. During its eighteen second duration, the video shows various elements of the PCR process including change in temperature and the role of the primer. However, it was commented by the team and by the clients that it was insubstantial in terms of information delivery, several of the stages of PCR are omitted with no mention of primer design, and in terms of interactivity.

Another video hosted on YouTube ('DNALearningCentre', 2010), currently referred to on School of Life Sciences' website, is similar in style to a lecture with slides and a voice-over which repeats the textual information on each slide. While this video is far more informative than the previous one, with each stage of PCR clearly described, and with visually pleasing animations, it lacks in explicit primer design and again in interactivity.

Finally, an animation from the University of Utah, titled "PCR Virtual Lab" (Genetic Science Learning Center, 2012). This is a much more interactive experience and allows the user to use virtual pipettes in order to simulate what you would do in a lab situation when performing PCR. Additionally, the information it provides, while slightly basic in the beginning for our target users, is extensive and very informative to the novice user, such as Biology-illiterate Computing Scientists. While this is a much more interactive and, compared to the alternatives described above, much more informative experience, it fails to provide the user with the theoretical background information, particularly on primer design, required to fully understand the process and why the reaction occurs.

Chapter 2

Design

2.1 Requirements

2.2 UI

Chapter 3

Implementation

3.1 UI

The implementation of the graphical user interface (GUI) required a number of decisions to be made before writing it could begin.

3.1.1 GUI Framework

Swing

Each member of the group had experience with the Swing framework, though not all of it good. The experience each member of the team had with Swing varied, and although every member had agreed that they had not liked using it in the past, we conceded that its integration with the Netbeans Integrated Development Environment (IDE), discussed in section 3.1.2, was extremely useful.

However, on investigating the framework more closely it was clear that Swing was extremely well documented with full API specification (Ora), and in-depth tutorials (swi). This was a huge part of our decision as we felt that the documentation provided would be more than adequate to allow us to use the framework with relative comfort.

JavaFX

Another framework considered was JavaFX which no member of the team had any experience with. Some members felt that this was a risk worth taking, given how much they disliked Swing, discussed above. In reality JavaFX was only briefly considered and totally disregarded when, upon brief investigation, JavaFX was still a relatively new framework, and was thus not quite as well documented as Swing, particularly when it came to troubleshooting on online forums.

In addition, JavaFX required Java 7, which, again, no member of the group had used before and which was not available, at the time, in the Level 3 Laboratory where we would be working for the majority of the year. It seemed like too much of a risk to try to learn two different technologies at the same time, while having to provide our own development platforms, which, with various members of the team never having used the Linux OS before, could potentially cause a number of problems.

Decision

The investigation was carried out by the group's Toolsmith, Ross Barnie, who presented the evidence discussed in the sections above regarding the two frameworks to the rest of the team. With this evidence the team voted in favor of using the Swing framework with Java 6.

Retrospectively, Swing, and Java 6, are out-of-date technologies and JavaFX is now packaged with Java 7 jav, so the application would have been more up-to-date or future-proof had we used JavaFX. Additionally, (some of) the computers in the level 3 lab now do have Java 7 installed upon another project team requesting it, so our fears over development platform problems were nullified, though this was only after we had started development.

It was an unfortunate shortcoming of the research into JavaFX that the group did not know about JavaFX's integration with the Netbeans IDE which was seen as one of the key differences between the two frameworks at the time of making the decision.

3.1.2 IDE

One concern was that, in some members' experience, using two separate IDEs was extremely time consuming, particularly while using version control. This was mostly due to various metadata that IDEs keep track of in various files, however this meant that any small change to the source code would change the metadata and therefore each commit would have to involve adding it, which would be very time-consuming.

It is because of this experience that the group decided to work from a single IDE, researched again by Ross Barnie.

Netbeans

Netbeans is an IDE which the team had had little experience with and had only used in the context of building applications with GUIs created using the Swing framework. There was some trepidation to using Netbeans since most of the team had associated their problems with Swing with Netbeans itself. Upon further research, which involved using the IDE to build small applications, Netbeans started much faster than Eclipse, discussed below. And the design interface was very simple and easy to use, with each element being laid out the way you wish and the associated source code being generated for you. This meant that the design layout could be finished very quickly, rather than spending our time writing hundreds of lines of source code just for the interface.

In terms of Netbeans' metadata, it was quite minimal and would not clutter the version control repository to an unacceptable degree.

Eclipse

The team had used Eclipse many times in the past, given that the entirety of the coursework for level 2 was to be done using it. Again, our experience of Eclipse is somewhat tainted by associations with problems we faced at the time, such as a bug on the version for Windows which meant that Eclipse would freeze if you tried to copy or paste anything.

The team felt, unanimously, that Eclipse was slow, not only to initially load, but also when trying to write code. Editing-wise, Eclipse was rather cumbersome and not much better than a text editor. Also the requirement to bind the “Workspace” was seen as a potential point for confusion and errors.

In addition, the team felt that the missing design interface seen on Netbeans, discussed above, was a huge disadvantage and would cause a significant loss of time, simply due to the volume of code we would have to write instead of being auto-generated.

Members of the team also pointed out that Eclipse has a tendency to create a large amount of metadata which would clutter the version control repository.

No IDE

It was briefly considered to have no IDE at all and simply use text editors. This would allow for extremely fast editing in a very comfortable environment, since most text editors, such as Vim or Emacs, are highly customisable and can launch in a matter of seconds. Text editors would also not require metadata, keeping our version controlled directories clean.

However, the obvious problem with no IDE is that troubleshooting problems becomes very tedious very quickly, and unlike IDEs, you cannot automatically import a missing package or method, nor can there be any auto-generated code for that matter.

Decision

When the evidence above was given to the team, we were also discussing which GUI Framework to use (as discussed in section 3.1.1) and it became obvious that integration with the framework would be key to helping us develop the GUI.

We therefore decided to work with the Netbeans IDE because of the design interface, minimal metadata, and lack of (known) bugs that would affect us in any meaningful way.

Retrospectively, this was the correct decision. Even if we had chosen a different GUI framework, the advantages of the easy-to-edit design interface far outweigh any problems we had with it.

3.2 Models, Custom Methods

Find a better title for this section.

Chapter 4

Evaluation

4.1 Testing

4.1.1 Demonstration

4.1.2 Questionnaire

Chapter 5

Conclusion

Chapter 6

Contributions

Bibliography

JavaFX overview. URL `http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm`.

Trail: Creating a gui with jfc/swing. URL `http://docs.oracle.com/javase/tutorial/uiswing/`.

Nick DeKanter. Gaming redefines interactivity for learning. *TechTrends*, 49(3):26–31, 2004.

User ‘DNALearningCentre’. Polymerase Chain Reaction (PCR), 2010. URL `http://youtu.be/XXkG6m3yT1M`.

User ‘dwildridge’. Taq Extension (test).wmv, 2012. URL `http://youtu.be/XXkG6m3yT1M`.

Genetic Science Learning Center. PCR Virtual Lab, August 2012. URL `http://learn.genetics.utah.edu/content/labs/pcr/`.

Java Swing API Summary. Oracle. URL `http://docs.oracle.com/javase/6/docs/api/javax/swing/package-summary.html`.