SGS: Splicing Graph Server

Durgaprasad Bollina[1], Bernett T. K. Lee[2], Shoba Ranganathan[1,2,*]
[1]Biotechnology Research Institute and Department of Chemistry & Bimolecular Science, Macquarie University, Sydney, NSW 2109, Australia.
[2]Department of Biochemistry, National University of Singapore, Singapore, 119260.
**To whom correspondence should be addressed: shoba@els.mq.edu.au*

ABSTRACT: SGS (Splicing Graph Server) is as web application based on the MVC architecture with a Java platform. The specifications of the implemented design pattern are closely associated with the specific requirements of splicing graphs for analyzing alternative splice variants from a single gene. The paper presents the use of MVC architecture using JavaBeans as a model, with a JSP viewer and the servlet as the controller for this bioinformatics web application, with the open source apache/tomcat application server and a MySql database management system.

1. INTRODUCTION

Alternative splicing is a ubiquitous process permitting single genes to encode many transcripts. Its disruption is associated with many diseases, such as cardiovascular, cancer and neurodegenerative disorders [1, 2, 5]. The number of genes in an organism does not seem to correlate with its complexity, suggesting a major role for alternative splicing in the production of diversity of the transcriptome [1]. Although the alternative splicing databases themselves provide an insight into the amount of alternative splicing, they do not provide any visualization representation and classification of the types of alternative splicing events occurring [3]. Splicing graphs help in representing transcript diversity and the classification of alternative splicing events which is critical for further work in deciphering the regulatory controls that govern alternative splicing [4]. In order that transcript information is useful to the experimental community, what is also needed is the representation of transcript diversity as single-line summary graph.

The gene structure information is first organized into transcript models and then the exons are clustered to identify distinct reference exons and variants, to form splicing graphs. This form of representation allows us to design simple rules that determine alternative splicing classifications [3]. Further we have designed a single-line splicing graph similar to the one used in experimental validation of alternative splicing events and PCR primer design. The splice graphs server and classification information are freely available at: http://sgs.biolinfo.org and to the best our knowledge there no such splicing graph server is available.

2. OVERVIEW OF THE WEB SERVER

Managing the user-interface has proven to be a difficult task for developers. There is substantial complexity in ensuring that information is presented accurately to users that users can control the application appropriately, and that views and domain objects are synchronized. To manage the complexity, reference architectures have been developed, such as Model-View-Controller (MVC) [6-8]. MVC architecture provides multiple views using the same model, easier support for new types of clients compared to CGI, clarity of design, efficient modularity, scalability and portability for ease of distributable.

In the case of a web application, MVC allows a web designer who has a different skill set to make changes to the interface without involving programming, where as in traditional CGI-script [9], every change to the website requires detailed knowledge of the logic of the program. With MVC, detailed knowledge is needed usually of only the relevant part of the system and even then for major changes alone [10]. MVC also allows the project to be broken down so that domain experts can work on their own parts.

Our interest in MVC architecture stems from our efforts in developing a Splicing Graph web server, for alternative splicing and transcriptome analysis. We are studying patterns for usability and considering their JAVA based implementations in building a robust bioinformatics web application. To this end we chose alternative splicing transcript diversity analysis approach with splicing graphs, on the basis that it would be a simple but effective way of representing transcript diversity using a novel visualization approach. This paper overcomes the problems associated with CGI based systems, applying MVC design patterns for bioinformatics web applications with its java implementations taking the splicing graph server as an example.

2. 1 Design Pattern

A design pattern describes a proven solution to a recurring design problem, placing particular emphasis on the domain model and enforces

connections with individual parts comprising the model. There are three main reasons to use design patterns. Firstly, they are proven so that the experience, knowledge and insights of developers who have used these patterns successfully in their own work can be accessed. Secondly, they are reusable. When a problem recurs, there is no need to invent a new solution; the pattern is followed and customized as necessary. Thirdly, they are expressive. Design patterns provide a common vocabulary of solutions, which can be used to express larger solutions easily, providing scalability.

It is important to remember that design patterns do not guarantee success. We can only determine whether a pattern is applicable by carefully reading its description. A successful pattern that has found many uses is Model-View-Controller (MVC), which was first implemented in Smalltalk (first object-oriented language) [6] in the 70's. Since that time, the MVC design idiom has become commonplace, especially in object-oriented systems.

2.2 MVC Architecture

The Model-View-Controller paradigm (Figure 1) is to separating the application object (model), the view (model representation to the user and the controller (how the user controls the dataflow). Models provide the core functionality of the system (e.g. a splicing graph object). Views present models to the users (e.g. all transcript information, splicing graph with references and distinct nodes and connections/single-line representation). There can be more than one view of the same model. Controllers control how the user interacts with the view objects and manipulate models or views (e.g. file upload, individual transcript view, splicing graph/single-line representation).

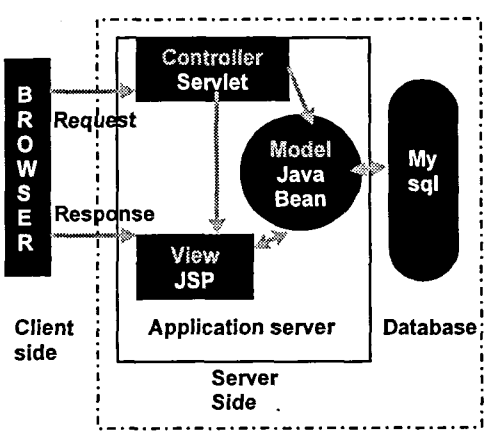

Figure 1. MVC architecture

The Model object stores the transcript data to be displayed. It also has operations that can be applied to transform, the transcript diversity information into splicing graphs using graph theory. However, it is ignorant of the user interface, the manner in which the data are to be displayed and data manipulation *via* user interface actions. Data access and manipulation are thus independent of the user interface.

The View object refers to the model. It uses the query methods to obtain data from the model and then displays the information. A view renders the contents of a model, in response to the user query. It accesses transcript information through the model and specifies how the graphical view should be presented. The View object maintains consistency in its presentation dynamically, while the data extracted from the model changes, in response to inputs from the user interface.

The Controller object provides the physical link for users to manipulate data within the model. A controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application (CGI/Perl based), they appear as GET and POST HTTP requests.

3. GRAPH THEORY AND SPLICING GRAPHS

The earliest paper on graph theory [11] dates back to 1786 by Leonhard Euler. Euler discussed whether or not it is possible to stroll around Konigsberg (later called Kaliningrad) crossing each of its bridges across the river Pregel (later called the Pregolya) exactly once, and gave the conditions which are necessary to permit such a stroll. Graph theory was developed further into directed graphs in 1856 by Thomas Pennyngton Kirkman and William Rowan Hamilton, who studied trips for visiting defined sites exactly once. Current examples of directed graphs include the World Wide Web, where files are the vertices or nodes and a link from one file to another is a directed edge (arc or connection).

3.1 Formal description for splicing graph

Let $T_1, T_2 \ldots, T_n$ be the set of all RNA transcripts of a given gene. Each transcript T_i corresponds to a set of genomic positions V_i. We define the set of all transcribed positions V, as the union of all sets V_i ($V \cup V_i$). The splicing graph G is the directed graph on the set of transcribed sites V that contains a directed edge (u, v) if and only if u and v are consecutive positions in one of the transcripts T_i. Every transcript can be viewed as a path in the splicing graph G, and the whole graph G is the union of all paths.

4. IMPLEMENTATION AND RESULTS

The SGS (Splicing Graph Server) enables the generation of dynamic images for visualization of different types of alternative splicing events by transforming transcript information into splicing graphs at the same website, following the identification of distinct reference exons. To create splicing graphs, you need to first create a GFF (General Feature Format) file containing sequence information organized as a series of exon position information. [12]. Data from the user in GFF format is converted into the transcript input model and passes to the server side program for dynamic image generation. The web server aims to provide the global research community with a free portal for splice graphs using GFF format and output options enabling the publication-quality images to be saved in various formats like png, gif, jpg, pdf with simple yet intuitive visualization of the intron–exon arrangement graphically. When a user uploads a sequence to SGS in GFF format, a data file is parsed and stored into the MySQL database on the server, using a Java bean. The stored data is converted into a splicing graph for dynamic display on the web browser on the client side.

Gene structure information including the location of the transcript and the start and end positions of each exon/intron that make up the transcript model are checked for consistency and then loaded into a list. The exon list is then retrieved and clustered on the basis of the exons occupying overlapping genomic positions. If an exon overlaps with another, the server checks which one has well-determined borders and also which one occurs in the majority of transcripts and then retains these as distinct exons. If an exon is completely contained in another larger exon, these are not merged but retained as individual exons and then entered into a list maintaining the mapping of variant exons to distinct exons. Splicing graphs are then drawn using these clusters of transcripts from the exon list. SGS follows the classification schema described in DEDB [3].

SGS is written entirely in Java, so that dynamic image generation is platform-independent. Server-side image generation gives us the ability to generate images with custom fonts, tables and graphs, which are impractical using HTML and JavaScript. Further, utilities are provided for the generation of high quality images in different formats. Both these advantages are especially powerful on the Java platform, using the Java Image API [13] (application program interface) and Servlets, compared to CGI script and other client server systems.

In order that transcript information is useful to the experimental community [14-18], we have designed a single-line graph, of the type that is seen in text molecular biology text books, by merging the all variant exons of distinct reference exons leading to single-line summary transcript diversity diagram. This one line graph is what an experimentalist will be interested for the gene of his/her alternative splicing events, as it suggests sequences for primer design across splice junctions. Further, alternative splicing events such as intron-retention are clearly seen in the single-line graph. SGS also transforms any anti-sense transcripts to the sense direction before calling the transcript model and provides a detailed list of classification events to help experimental verification of alternative splicing. An example of the splicing graph and single-line diagram for a human gene with six transcripts is shown in Figure 2, which clearly shows the utility of the splicing graph approach over the conventional representation of each transcript as a separate lines, used in the ASD database (5).

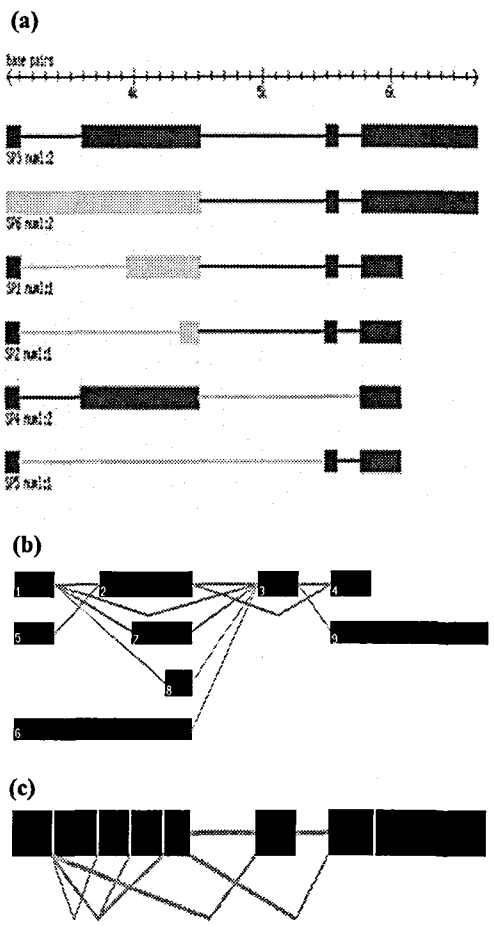

Figure 2. Transcript diversity information for the human hyaluronidase gene, HYAL1 (a) from ASD (5) and from SGS as (b) a splicing graph and (c) a single-line splice diagram.

5. FUTURE WORK

Future work will focus on integrating additional relevant information onto the splicing graph

server such as different classification scheme for splicing events, location of specific splice sites regions (examples GC-rich regions) and the comparison of alternative splicing events in different genomes by representing them as splice graphs.

6. ACKNOWLEDGEMENT

DP is grateful to the Macquarie University for the award of a research scholarship (iMURS) and a travel grant (MUPGR).

7. REFERENCES

[1] Pennisi E. Why do humans have so few genes? *Science.* 309:80. 2005.
[2] Graveley,B.R. Alternative splicing: increasing diversity in the proteomic world. *Trends Genet.* 17, 100–107. 2001.
[3] Lee BT, Tan TW, Ranganathan S. DEDB:a database of Drosophila melanogaster exons in splicing graph form, *BMC Bioinformatics* , 5:189.2004.
[4] Heber S, Alekseyev M, Sze SH, Tang H, Pevzner PA. Splicing graphs and EST assembly problem. *Bioinformatics 2002,* 18 Suppl1:S181-8. 2002.
[5] Thanaraj TA, Stamm S, Clark F, Riethoven JJ, Le Texier V, Muilu J. ASD: the Alternative Splicing Database. *Nucleic Acids Res,* 32 Database issue:D64-9. 2004.
[6] Howard T. The Smalltalk developer's guide to VisualWorks. SIGS, New York, USA. 1995.
[7] Krasner GE and Pope ST. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *JOOP,* 1:26–49. 1988.
[8] Shan Y. Mode: A uims for Smalltalk. In N.Meyrowitz, editor, *ECOOP/OOPSLA '90 Proceedings*, 258–268.ACM Press, New York. 1990.
[9] Kim E. CGI-Developers-Guide. Sams Publishing, Indianapolis, USA. 1996.
[10] Sun Java Design patterns http://java.sun.com/blueprints/patterns/MVC.htm
[11] Biggs NL, Lloyd EK and Wilson RJ. Graph Theory 1736-1936. Clarendon Press, Oxford, UK. 1976.
[12] GFF (General Feature Format). http://www.sanger.ac.uk/Software/formats/GFF/
[13] Java Image API. http://java.sun.com/j2se/1.4.2/docs/api/
[14] Leipzig J, Pevzner P, Heber S. The Alternative Splicing Gallery (ASG): bridging the gap between genome and transcriptome. *Nucleic Acids Res,* 32:3977-3983. 2004.
[15] Lee C, Atanelov L, Modrek B, Xing Y ASAP: the Alternative Splicing Annotation Project. *Nucleic Acids Res,* 31:101-105. 2003.
[16] Black,D.L. Protein diversity from alternative splicing: a challenge for bioinformatics and post-genome biology. *Cell,* 103, 367–370. 2002.
[17] Lopez AJ. Alternative splicing of pre-mrna: developmental consequences and mechanisms of regulation. *Annu. Rev.Genet.,* 32:279-305. 1998.
[18] Clark F. and Thanaraj T.A. Categorization and characterization of transcript-confirmed constitutively and alternatively spliced introns and exons from human. *Human Molecular Genetics* 11: 451-464 . 2002.