Defining Mapping Mashups with BioXMash

Ela Hunt[1], **Joanna Jakubowska**[2], **Caroline Bösinger**[1], **and Moira C. Norrie**[1]

[1]Department of Computer Science, ETH Zurich, CH-8092 Zurich
[2]Department of Computing Science, University of Glasgow, Glasgow, UK
{hunt,Norris}@inf.ethz.ch, asia@dcs.gla.ac.uk

Summary

We present a novel approach to XML data integration which allows a biologist to select data from a large XML file repository, add it to a genome map, and produce a mapping mashup showing integrated data in map context. This approach can be used to produce contextual views of arbitrary XML data which relates to objects shown on a map. A biologist using BioXMash searches in XML tags, and is guided by XML path data availability, shown as the number of values reachable via a path, in both global, genome-wide, and local, per-gene, context. Then she examines sample values in an area of interest on the map. If required, the resulting data is dumped to files, for subsequent analysis.
This is a lightweight integration approach, and differs significantly from other known methods. It assumes that data integration can be performed on a lab computer with limited memory, with no database installation or programming knowledge. It is different from BioMarts which predefine possible data selections, in that arbitrary data sources related to map items can be used. BioXMash offers full textual search in XML paths, shows path statistics, and supports visual verification of data values. Repeated scanning of all XML files at query time is avoided by the use of a high level indexing technique. Our prototype demonstrates this new approach. It efficiently supports data browsing on 2 GB od data from GeneCards with an index size of 40 MB.

1 Introduction

Integrative biology gains new knowledge by combining existing insights from many biological subdisciplines. The process of integrative data analysis uses at least four recognised strategies [8]: data transformation, typology development, extreme case analysis, and consolidation/merging. In our domain data transformation includes modelling, programming and statistical analysis techniques which deliver data and statistics in a suitable format, or visualisations which support direct analysis. Typology development is supported by abstractions, such as promoter boxes or introns and exons. Extreme case analysis is a way of arguing a point by discussing a vivid example which best convinces the audience, as exemplified by this paper. Finally, consolidation/merging is a very important part of research which can be achieved by a combination of data transformation and visualisation. This last strategy is supported in bioinformatics by a variety of techniques, and both genome and pathway maps are of great help. However, we still do not have good tools which strongly support data consolidation and merging, and addition of new data to maps. This gap is the focus of our work, and BioXMash addresses the researcher's need to integrate data on the fly, ideally with zero programming effort.

Map mashups[1] are interactive web applications combining content from multiple sources into a new page which superimposes various types of data on a map [20, 15]. Genome data for a mashup can be served by a Distributed Annotation Server (DAS) [4] and can be shown in an Ensembl map [10]. However, DAS does not offer information integration, and is not trivial to set up. The difficulty of data integration lies in: (1) finding the relevant data type (such as a probe sequence, peptide sequence, DNA motif, or reference to another database) in a large XML schema or large XML repository, such as offered by GeneCards [24, 25] or ArrayExpress [3]; (2) selecting a subset of schema or data that fits our map and is related to our needs; and (3) integrating this data with our chosen map, that is matching annotations and items to map positions or identifiers and displaying them in context. Currently, no tools can support visual data integration without extensive programming. We believe that such techniques are indispensable in information consolidation, and we are developing a technology that can generate them with minimum user effort.

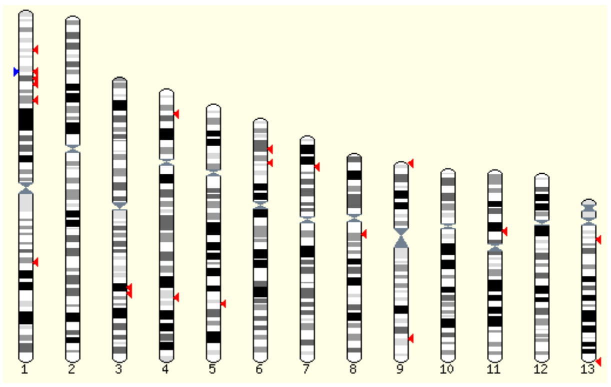


Figure 1: Ensembl: a probe (triangle) on several human chromosomes. The map provides context for this probe, and shows that it is not unique.

Ensembl supports a basic functionality of showing annotations on a map, see Ensembl KaryoView at www.ensembl.org, see Fig. 1. Fischer et al. [6] visualise both probes and eQTLs (quantitative trait loci defined by gene expression) shown in karyotype context. In supporting hypertension research we regularly create mashups which add the results of micro array experiments conducted in the lab, or derived by other groups [13], see Fig. 2. One can enhance a genome map in several ways, by either adding new object types to the display, such as QTLs, or by adding textual annotations to the existing items, or adding colours or markers. In that way, genome or pathway maps can be enriched to support our understanding of data relationships.

Integrated views of data are required in candidate gene analysis where we focus on long QTLs and use a combination of functional genomic approaches [17, 18]. However, programmatic construction of such integrated perspectives remains beyond the reach of most biologists. The aim of our work is to create new data integration technology for lab use. Our contributions are as follows. We present a toolset which uses a new browser, VisGenome [14], and combines it with a new XML indexing solution, to produce mashups interactively, without programming. We outline a use case for these tools, report on the index, the query execution strategy, and on system implementation.

The paper presents a use scenario and system requirements in Section 2, and follows on with a

[1]en.wikipedia.org: Mashup, or bootleg, is a musical genre which, in its purest form, consists of the combination of the music from one song with the a cappella from another.

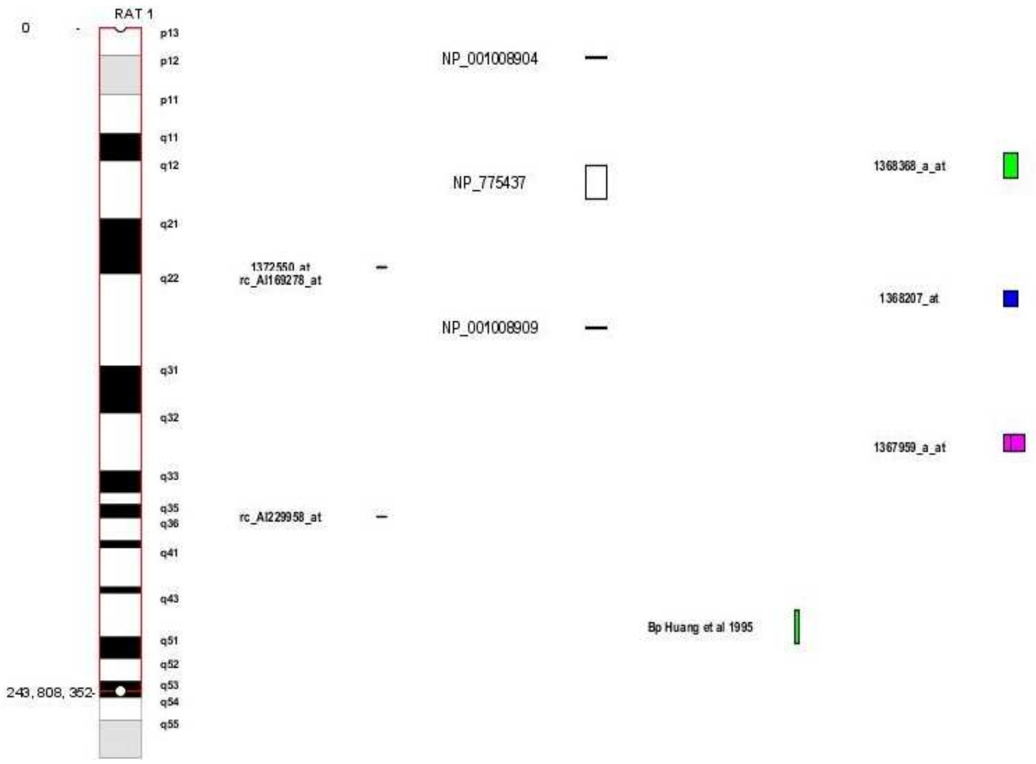


Figure 2: Visualisation of integrated data in VisGenome. Rat chromosome 1 is shown, and left to right: microarray probes and genes; further to the right, in colour: QTLs and experimental results added from other sources.

justification from the HCI perspective (Section 3). Section 4 presents the index and the queries it supports, and Section 5 focuses on the architecture and implementation details. Section 6 summarises related work and Section 7 offers a discussion and conclusions.

2 A Mashup Scenario and System Requirements

Figure 3 shows an example use scenario. The user performs the following actions.

1. She *identifies a map*, for example an Ensembl map of human chromosome 21.
2. She decides which of the *types of data* shown on the map to enhance with additional information. For example, she decides to add Gene Ontology, www.geneontology.org, (GO) annotations for each gene.
3. Alternatively, she *defines new types of data* for which information will be found in XML files, for instance QTLs or some experimental results, see Fig. 2. We assume that for each data item map coordinates are given.
4. The user *identifies XML sources* for integration, either on her computer, or on the web. She triggers indexing locally or remotely, or downloads an index from a web site. For example, she decides to use GeneCards [25, 24].

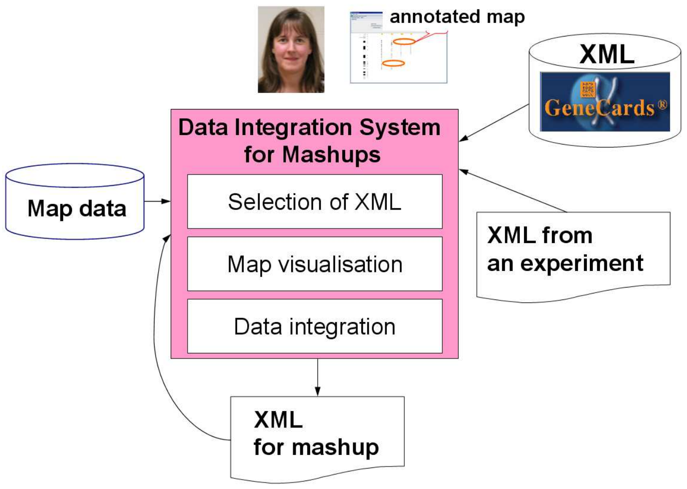


Figure 3: Mashup construction. A biologist interacts with a visualisation which imports map data, and selects XML paths from files or databases. XML data corresponding to the selected paths supports data integration and visualisation. The user then studies an annotated map.

5. The biologist queries for a term or a combination of terms, using regular expressions, such as "swissprot*protein" or "GO", see Fig. 4. She examines the indexed data summaries and sees that in GeneCards there are 6 XML paths representing GO terms, and that 24,907 genes have a cellular component annotation, 35,179 have a biological process annotation, and 41,985 have a molecular function annotation. She sees that the paths for GO terms are related, as they have a shared prefix. The user ticks all the boxes associated with GO terms.

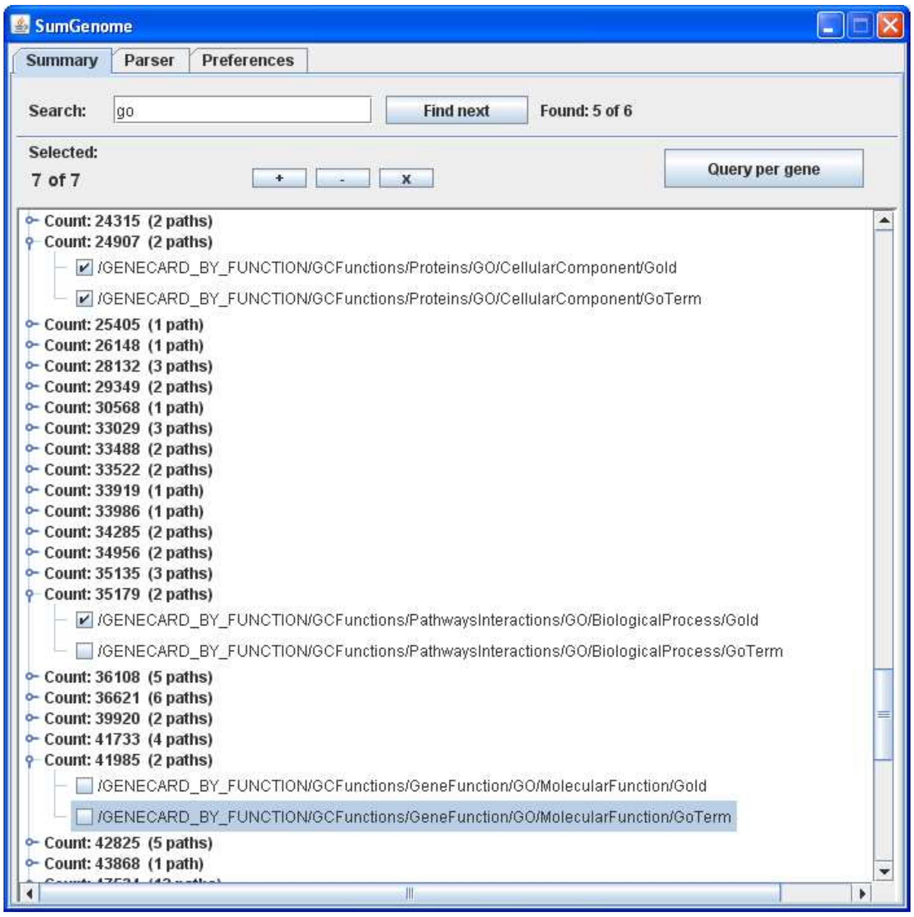
6. She examines the content of paths she selected, see Fig. 5. In one of the areas of interest on human chromosome 21, one of the genes includes a cellular component "intermediate filament". This confirms that this is the data she wants to add to the map, and will later be shown, by colouring each gene with its function. The biologist also notices that there are many genes which do not have any GO annotations and may abandon this data integration attempt for that reason.

BioXMash supports searching in XML paths and shows how many values are available for each path globally, and for each gene. It also executes XML queries on demand. The use scenario is made possible by the following system functions.

1. Index creation, creating a mapping from XML to a summary data structure.

2. Efficient query execution, to support the user in finding interesting paths and verifying data values in a large XML data set. The system will execute tree projections, based on selected paths, and, indirectly, selections based on leaf values.

3. Efficient joining of XML with the map. This is based either on (1) comparing XML file names with gene names, as implemented in BioXMash, (2) comparing leaf values reachable via the selected path with identifiers on the map, or (3) checking mapping coordinates provided in the XML file against the map, to select for instance the data for the given chromosome only, or (4) other user-defined criteria, such as sequence match.



Search: go

Found: 5 of 6

Selected: 7 of 7

```
Count: 24315 (2 paths)
Count: 24907 (2 paths)
    [✓] /GENECARD_BY_FUNCTION/GCFunctions/Proteins/GO/CellularComponent/GoId
    [✓] /GENECARD_BY_FUNCTION/GCFunctions/Proteins/GO/CellularComponent/GoTerm
Count: 25405 (1 path)
Count: 26148 (1 path)
Count: 28132 (3 paths)
Count: 29349 (2 paths)
Count: 30568 (1 path)
Count: 33029 (3 paths)
Count: 33488 (2 paths)
Count: 33522 (2 paths)
Count: 33919 (1 path)
Count: 33986 (1 path)
Count: 34285 (2 paths)
Count: 34956 (2 paths)
Count: 35135 (3 paths)
Count: 35179 (2 paths)
    [✓] /GENECARD_BY_FUNCTION/GCFunctions/PathwaysInteractions/GO/BiologicalProcess/GoId
    [ ] /GENECARD_BY_FUNCTION/GCFunctions/PathwaysInteractions/GO/BiologicalProcess/GoTerm
Count: 36108 (5 paths)
Count: 36621 (6 paths)
Count: 39920 (2 paths)
Count: 41733 (4 paths)
Count: 41985 (2 paths)
    [ ] /GENECARD_BY_FUNCTION/GCFunctions/GeneFunction/GO/MolecularFunction/GoId
    [ ] /GENECARD_BY_FUNCTION/GCFunctions/GeneFunction/GO/MolecularFunction/GoTerm
Count: 42825 (5 paths)
Count: 43868 (1 path)
```

Figure 4: Paths grouped by the number of values present in data, with user searching for "go". The user selects a path by ticking the box next to path name.

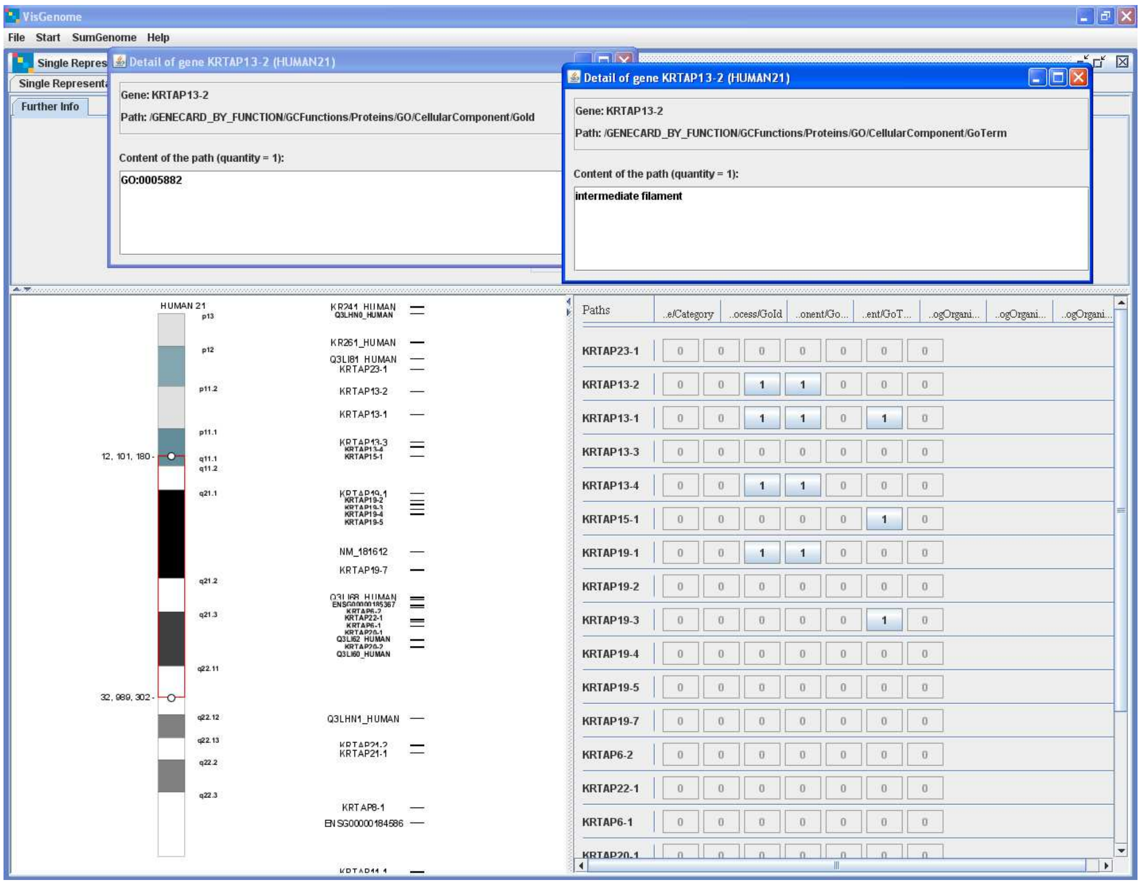
4. *Querying of the selected data* in map context, to show data values, see Fig. 5.

5. Visualisation of the selected annotations, done by the mapping package.

6. *Generation of output files* recording the queries, a user log, statistics, and an XML dump of the selected data.

The scenario poses *several challenges*. First, we need to support the user in *selecting the data for a mashup*. This could be delegated to an existing schema viewer, or an XML file viewer, but there are several reasons not to use such a viewer, or at least to enhance it with additional information. The schema is large, often fills in several screens, and the schema on its own does not provide enough information, as data semantics are only clear when the data values are seen. The schema does not tell us how much data is available, either. There is usually too much data to view, and it is hard to select the right files for viewing in a repository with thousands of files. In some cases the entire XML is just one file, and is too large to open. The problem of data selection is solved via visualisation and summarisation.

The second problem is the *efficiency of data browsing and querying*. GeneCards [24, 25] integrates data programmatically from over 100 sources and offers over 2 GB of data (April 2006), and the latest release is significantly larger. Our system must support interactive data selection without scanning all data for each query, as a biologist's computer does not normally house a



Detail of gene KRTAP13-2 (HUMAN21)

Gene: KRTAP13-2

Path: /GENECARD_BY_FUNCTION/GCFunctions/Proteins/GO/CellularComponent/Gold

Content of the path (quantity = 1):

GO:0005882

Detail of gene KRTAP13-2 (HUMAN21)

Gene: KRTAP13-2

Path: /GENECARD_BY_FUNCTION/GCFunctions/Proteins/GO/CellularComponent/GoTerm

Content of the path (quantity = 1):

intermediate filament

Paths	..e/Category	..ocess/GoId	..onent/Go...	..ent/GoT...	..ogOrgani...	..ogOrgani...	..ogOrgani...
KRTAP23-1	0	0	0	0	0	0	0
KRTAP13-2	0	0	1	1	0	0	0
KRTAP13-1	0	0	1	1	0	1	0
KRTAP13-3	0	0	0	0	0	0	0
KRTAP13-4	0	0	1	1	0	0	0
KRTAP15-1	0	0	0	0	0	1	0
KRTAP19-1	0	0	1	1	0	0	0
KRTAP19-2	0	0	0	0	0	0	0
KRTAP19-3	0	0	0	0	0	1	0
KRTAP19-4	0	0	0	0	0	0	0
KRTAP19-5	0	0	0	0	0	0	0
KRTAP19-7	0	0	0	0	0	0	0
KRTAP6-2	0	0	0	0	0	0	0
KRTAP22-1	0	0	0	0	0	0	0
KRTAP6-1	0	0	0	0	0	0	0
KRTAP20-1	0	0	0	0	0	0	0

Figure 5: A map if human chromosome 21 showing genes. To the right of the genes, a panel of boxes shows path statistics for each selected path (column) and gene (row). Clicking on buttons showing counts fires queries which deliver XML values for each gene/path combination, shown above the map panel. This allows the user to verify her understanding of the information encoded by the XML paths she selected and evaluating data coverage.

robust database system or have enough RAM to support fast queries on this scale. It is clear that indexing is required, and the goal is to find a minimal index that can support these requirements.

3 HCI Justification for the Index

We aim to support the user in selecting data for addition to the map. When the XML document is large, it is not easy to understand such a large variety of information. Using a tree representation of XML does not produce a satisfactory outcome, as one only sees a small part of the tree at any time, say 20 to 30 paths, and the same is true for hyperbolic viewers [19]. Schema understanding is hard even for an experienced data engineer, so alternatives are called for. One could use an ontology to mediate user choice [1], or a high level summary [27]. Since an appropriate ontology or an abstract schema is not available in most cases, we propose a simpler solution, an integration index. Recent work in HCI clearly shows that summarisation is an important tool and can make a query interface more useful and efficient [26]. This is one of the main reasons for our adoption of simple path summaries. In addition to statistics, we group paths by count and support lexical searches in paths.

4 Index Definitions and Queries

We propose a new type of index for this task, a *data integration index* which supports interactive browsing. The index provides both a high-level overview of the data, and fast query access, in order to show data values to the researcher.

The index consists of three components: a global index, a full text index to the text of all paths in the tree, and a gene-path index. To illustrate our solution, we present an XML fragment, a tree projection from GeneCards, showing a gene name and its map position.

```
<GENECARD_BY_FUNCTION xmlns:xsi="http://www.w3.org/2001/XMLSchema-in-
 stance" xsi:noNamespaceSchemaLocation="GeneCardByFunction.xsd">
  <Header><GeneSymbol><SYSTEM><Symbol>
   3.8-1
  </Symbol></SYSTEM></GeneSymbol></Header>
  <GenomicLocation><Cyto><ENTREZGENE><CytoBand>
   6p21.3
  </CytoBand></ENTREZGENE></Cyto></GenomicLocation>
</GENECARD_BY_FUNCTION>
```

This fragment defines the location of gene `3.8-1` to be `6p21.3`. It contains the following two paths p which point to the leaves containing the gene name and its location.

```
p 1. /GENECARD_BY_FUNCTION/Header/GeneSymbol/SYSTEM/Symbol
p 2. /GENECARD_BY_FUNCTION/GenomicLocation/Cyto/ENTREZGENE/CytoBand
```

The index, $Summary = (GI, TI, GPI)$, consists of three relations.

GI: GLOBALINDEX lists all distinct paths p in an XML repository, and the counts of their occurrences, $GI : (p, count(p))$. In our example the index contains tuples (1,1) and (2,1), as each path is present once. GI allows the user to assess the statistical relevance of data reachable by a path, and to access the XML path text via path ID (here IDs are 1 and 2). If no data items can be reached by a path, the user will ignore it. If there are a lot of data, the user can verify data values. If the counts for two paths are similar, and the paths share a prefix, the paths may be parts of a larger concept, which may then be examined by a user. An extended index may additionally contain a typical value for each path, a histogram, or a data type.

TI: TEXTINDEX is a full text index to paths, based on tokenisation, $TI : (p, tokens(p))$. The entry for path 1 would be (1, (GENECARD_BY_FUNCTION, Header, GeneSymbol, SYSTEM, Symbol)). TI allows the user to find paths of interest, and supports regular expression queries.

GPI: GENEPATHINDEX indexes the combination of paths and XML files, *XMLfile*, or partitions. Where the XML data consist of many small files, the index maps each path and file combination to the count of path in a file, $GPI : (p, XMLfile, count(p))$. This is based on an assumption, made in GeneCards, that each file describes one distinct object, for instance a gene, microarray probe, or a QTL. Where XML is a large monolithic file, the user may have to guide the system in how to discern parts and limit query scope.

While GI and TI summarise the XML at an overview level, GPI supports the user in examining the data in detail, and mediates access to XML files or fragments.

```
declare variable $doc as node() external; // query GI
<results> {
   for $quant in fn:distinct-values($doc//mapping/quantity)
   order by fn:number($quant) ascending
   return <result>
           <quantity>{ $quant }</quantity> {
           for $m in $doc//mapping
           where $m/quantity = $quant
           order by $m/path ascending
           return <pathMapping> { $m/path } </pathMapping>
           }
         </result>
}
</results>
```

Figure 6: Query returning paths ordered by count.

```
<map>
    <type> gene </type> <name> 3.8-1 </name>
    <chromosome> 6 </chromosome>
    <start> 31,985,270 </start> <end> 33,584,532 </end>
</map>
```

Figure 7: Mapping information in XML format.

GI supports the execution of a query producing groups of paths ordered by the number of times a path occurs, see Fig. 6. GPI has two distinct roles. First, it shows the user the number of times a path occurs in the XML file corresponding to a map object, shown in Fig. 5 as little boxes with numbers, each box representing one of the selected paths for a gene of interest. Second, it supports queries on an *XMLfile* corresponding to the map object. The query is a selection of all values for a given path in the *XMLfile*. GI is used to locate the required files, and each file is read while the path query is being evaluated. This query runs fast enough not to require additional indexing support.

We now look at alternative strategies of joining XML with the map. The map visualisation system offers an API delivering a list of object names on the map and their positions, which can be captured for our example gene, as shown in Fig. 7. The following join strategies are possible.

1. Hardcoded: a predefined join is made between stripped *XMLfile* names and map identifiers in $/map/name$, executed at run time. We match substring ‘gene’ of file name, CARD_GENE.XML to values in $/map/name$.

2. Candidate keys could be identified for the XML data, as shown in [21], or declared by the data provider. Using keys allows one to attempt a join between each key path and $/map/name$.

3. The most time-intensive but flexible solution is to try joining on any of the paths selected by the user to $/map/name$. This may be useful where the user does not want to guess the semantics of data, and wants to explore some of the available annotations to see if

they are compatible with her map. Optionally, an index mapping values in $/map/name$ to paths selected by the user can be constructed $mapindex : (nameValue, p, count(p))$ to support the exploration of possible joins.

5 BioXMash Architecture and Implementation

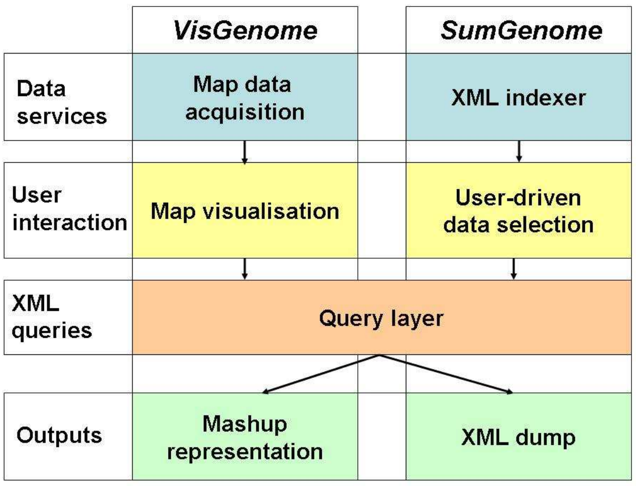


Figure 8: BioXMash is composed of VisGenome and SumGenome. Both applications provide a data service, a user interaction component, and generate outputs. They use a shared XML data integration component which performs queries.

Figure 8 shows the architecture of BioXMash. VisGenome [14] provides map data acquisition and visualisation, while SumGenome manages XML data. The components communicate via XML queries to execute a join between the map and the XML paths selected by the user. VisGenome provides zoomable genome maps for the human, the mouse and the rat. It downloads data from Ensembl via JDBC. The data consist of several vectors, each listing objects of a particular type (for instance gene) with coordinates and textual annotation. VisGenome shows textual annotations in a panel which is linked to the data the user mouse is positioned over. VisGenome can also add new types of objects to the map.

In the experiment we used GeneCards, XMLbyFunc, Apr 20 2006, version 2.34XMLbeta6. This data contains over 48 thousand files, totalling 2 GB, showing 420 XML paths. BioXMash implementation uses: Nux, *dsd.lbl.gov/nux*, for XQuery execution; Lucene, *lucene.apache.org*, for access to text in XML paths; and Java 1.5 and Java Serialization to store the indexes persistently. The machine was a Pentium 4 PC, with 3.4 GHz and 512 MB RAM. Index creation required a Java heap of 128 MB. Index creation was timed three times, taking for SAX, *java.sun.com/j2se/1.5.0/docs/guide/xml/jaxp* between 10.2 and 11.4 minutes, and for the PULL parser, *www.xmlpull.org*, between 10.7 and 11.6 minutes, so both methods performed similarly. GPI size is dominated by the array of size $|D| \times |P|$ where D are the genes (here XML files, one per gene) and P are paths. For over 400 paths and 48,000 genes, GPI is about 40 MB, using a serialised 2-D array of small integers (2 Bytes), and two arrays indexing files and paths.

Once the XML is indexed, the system supports interactive viewing of the entire data set, and performs satisfactorily. We used it for data semantics exploration in the construction of a schema matching benchmark [5]. In that exercise we compared map views generated by our software with the content of GeneCards, Swiss-Prot (expasy.org/sprot), and Ensembl. This helped us to verify our mappings.

6 Related Work

The challenge of user-driven query definition was first addressed by Zloof [28]. A similar approach was reported for XQuery [2], and is implemented in some biological datamarts [16]. Recently, ^{my}Grid [22] reformulated this challenge into the user-driven construction of workflows. We have not been able to find many user studies evaluating the effectiveness of those approaches. Recently, Tanin and colleagues showed that generalised query previews should be used in some contexts to help with query formulation [26]. Such previews offer data statistics, and shorten the time of query construction if the user is new to a data source and the query task is not very precisely stated. Tanin's work has motivated the use of statistics in BioXMash.

Most of the related work in XML indexing focuses on data structures which support XPath or XQuery evaluation. Our index has a different purpose, as we do not think a biologist will pose complex queries on data they are new to and want to explore. The index is closest in spirit to DataGuides [7], while current indexing developments like XSketch [23] have a completely different purpose, produce larger indexes, and will support complex database queries and not user-driven data integration on a database agnostic computer.

VisGenome [14] is one of many possible browsers that can be used for mashup construction. We previously reviewed a number of browsers [12], and carried out a user study comparing Ensembl and VisGenome [13]. Further background can be found in [11].

7 Discussion and Conclusions

BioXMash is a new data integration tool which collaborates with a visualisation application. It uses three simple data structures, serialised on disk, to allow a biologist to traverse XML and select the data that they want to analyse visually. We were inspired in this work by previous data integration approaches, in particular Clio [9] which offers query result previews. We have also drawn inspiration from user studies which show that data summarisation can help in data querying where the users are new to the data [26]. It is still unclear how usable our technology is, and we will investigate this aspect shortly. Our work in schema integration [5] showed that the system was helpful in identifying mappings between GeneCards and other biological data sets.

The production of a data integration index takes some time. We believe that this overhead might be absorbed by the data provider who could export such an index together with the XML schemas and data. Ideally, a web service hosted by a data provider should serve the data, and an index. This would go someway towards enabling large scale data mining, by exporting high-level data attributes which will optimise query construction, and enable more focused queries.

We believe the index needs to be further compressed. Various forms of data integration indexes can be tested, and various compression methods tried, to minimise the size of the index, and shorten both index transmission time, and more importantly, reduce RAM requirements on the user machine, as the index is cached at query time. Our system does not directly support value queries, as a full text index would be needed for that. This is an area for future work.

We can currently show all the annotations, but are still improving user interaction and display legibility in our toolset. We do not show data histograms, but would like to test such an option.

The main shortcoming of our system is that we do not know yet what types of data summaries and summary representations will be popular with users and how effective this approach will be. Future work will have to address challenges of the correct choice of statistical functions a biologist can intuitively grasp and use, and the representation of such data.

CONCLUSION We presented BioXMash, a new system enabling direct integration of XML data in a genome map. Our prototype is based on small indexing structures which support efficient XML queries but do not index data values. The system was tested with XML from GeneCards, and mapping data from Ensembl, and performed well in interactive data analysis. We are now ready for a user study with GeneCards and other XML datasets.

ACKNOWLEDGEMENTS This work was funded by an EU Marie Curie fellowship to E.H. C.B. carried out this project as part of her Diploma in the GlobIS group. J.J. is funded by the Medical Research Council, UK (grant to E.H. and M. Chalmers). We thank M. Safran of the Weizmann Institute of Science for providing a copy of GeneCards, and for her comments. We also thank the referees for their input.

References

[1] P. G. Baker et al. TAMBIS - Transparent Access to Multiple Bioinformatics Information Sources. In ISMB, pages 25–34, 1998.

[2] D. Braga et al. XQBE: a visual environment for learning XML query languages. In SIGMOD, pages 903–905, 2005.

[3] A. Brazma et al. ArrayExpress – a public repository for microarray gene expression data at the EBI. Nucleic Acids Research, 31(1):68–71, 2003.

[4] R. D. Dowell et al. The distributed annotation system. BMC Bioinformatics, 2:7, 2001.

[5] F. Duchateau et al. XBenchMatch: a Benchmark for XML Schema Matching Tools In VLDB, 2007. demo, to appear.

[6] G. Fischer et al. Expressionview: visualization of quantitative trait loci and gene-expression data in Ensembl. Genome Biology, 4:Research77, 2003.

[7] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimizaton in Semistructured Databases. VLDB, 436-445, 1997.

[8] J. C. Greene et al. Toward a Conceptual Framework for Mixed-Method Evaluation Designs. Educational Evaluation and Policy Analysis, 11(3):255–274, 1989.

[9] M. A. Hernandez et al. Mapping XML and Relational Schemas with Clio. In ICDE, pages 498–499, 2002.

[10] T. Hubbard et al. The Ensembl genome database project. Nucleic Acids Research, 30:38–41, 2002.

[11] E. Hunt, et al. The Visual Language of Synteny. OMICS, 8(4):289–305, 2004.

[12] J. Jakubowska, et al. Granularity of genomics data in genome visualisation. Technical Report, University of Glasgow, Department of Computing Science, 2006. www.dcs.gla.ac.uk/publications/paperdetails.cfm?id=8212

[13] J. Jakubowska et al. Usability of VisGenome and Ensembl – A User Study. Technical Report, University of Glasgow, Department of Computing Science, 2007. www.dcs.gla.ac.uk/publications/paperdetails.cfm?id=8510.

[14] J. Jakubowska et al. VisGenome: visualisation of single and comparative genome representations. Bioinformatics, to appear. www.dcs.gla.ac.uk/∼asia/VisGenome.

[15] A. Jhingran. Enterprise information mashups: Integrating information, simply. In VLDB, pages 3–4, 2006.

[16] A. Kasprzyk et al. EnsMart: a generic system for fast and flexible access to biological data. Genome Res., 14(1):160–9, 2004.

[17] M. W. McBride et al. Microarray analysis of rat chromosome 2 congenic strains. Hypertension, 41:847–853, 2003.

[18] M. W. McBride et al. Functional genomics in hypertension. Curr Opin Nephrol Hypertens., 15(2):145–51, 2006.

[19] T. Munzner. H3: laying out large directed graphs in 3d hyperbolic space. In INFOVIS, pages 2–10, 1997.

[20] S. Murthy et al. Mash-o-matic. In DocEng, pages 205–214, 2006.

[21] T. Pankowski and E. Hunt. Data Merging in Life Science Data Integration Systems. In IIS, pages 279–288. Springer, 2005.

[22] S. Pettifer et al. myGrid and UTOPIA: an integrated approach to enacting and visualising in silico experiments in the life sciences. In DILS, pages 59–70. Springer, 2007.

[23] N. Polyzotis and M. N. Garofalakis. XSKETCH synopses for XML data graphs.. ACM Trans. Database Syst., 31(3):1014–1063, 2006.

[24] M. Safran et al. GeneCards 2002: towards a complete, object-oriented, human gene compendium. Bioinformatics, 18(11):1542–1543, 2002.

[25] M. Safran et al. Human Gene-Centric Databases at the Weizmann Institute of Science: GeneCards, UDB, CroW 21 and HORDE. Nucleic Acids Res., 31(1):142–6, 2003.

[26] E. Tanin et al. Browsing large online data tables using generalized query previews. Inf. Syst., 32(3):402–423, 2007.

[27] C. Yu and H. V. Jagadish. Schema summarization. In VLDB, pages 319–330, 2006.

[28] M. M. Zloof. Query-by-example: the invocation and definition of tables and forms. In VLDB, pages 1–24, 1975.